Roadmap on Control of Real-Time Computing Systems

Control for Embedded Systems Cluster

EU/IST FP6 Artist2 NoE

Contents

1. Executive Overview on Control of Real-Time Computing Systems . . . 3

1.1 Motivation and Objectives . . . 3

1.2 Background . . . 3

1.3 Feedback scheduling . . . 4

1.4 Important issues . . . 5

1.5 Control of queues . . . 8

2. Current Industrial Practice and Needs . . . 9

3. Control of Servers . . . 10

3.1 Control-based queue admission control . . . 10

3.2 Control-based queue delay control . . . 13

3.3 Relative Queue Delay Control . . . 15

3.4 Control of Real-Time Databases . . . 16

3.5 Challenges and Research Directions . . . 17

4. Control of CPU Resources . . . 19

4.1 Feedback-Based Task Scheduling . . . 20

4.2 Feedback-based Bandwidth and Reservation Allocation . . . 21

4.3 Challenges and Research Directions . . . 22

5. Feedback Scheduling of Control Systems . . . 23

5.1 Actuators & Sensors . . . 24

5.2 Optimal Stationary Feedback Scheduling . . . 25

5.3 Optimal Feedback Scheduling . . . 26

5.4 Feedback Scheduling Structures . . . 27

5.5 Value-Based Feedback Scheduling . . . 27

5.6 Research Directions . . . 27

6. Control Middleware . . . 28

6.1 Research Directions . . . 30

7. Control of Communication Networks . . . 31

7.1 Congestion Control . . . 31

7.2 Control and optimization of wireless networks . . . 33

7.3 Interactions between TCP and wireless links . . . 34

7.4 Challenges and Research Directions . . . 36

8. Error Control of Software 37
8.1 Research Directions 38
9. References 39

1. Executive Overview on Control of Real-Time Computing Systems

1.1 Motivation and Objectives

An important result of the EU ARTIST FP5 project was four roadmaps on Hard Real-Time Development Environments, Component-Based Design and Implementation Platforms, Adaptive Real-Time Systems for Quality of Service Management, and Execution Platforms respectively, [Bouyssounouse and Sifakis, 2005]. The current roadmap written by the partners of the Control for Embedded Systems cluster within the EU/IST FP6 Network of Excellence ARTIST2 can partly be viewed as an extension of the adaptive real-time system roadmap. The focus is how flexibility, adaptivity, performance and robustness can be achieved in a real-time computing or communication system through the use of control theory.

Similar to the ARTIST roadmaps this roadmap is intended as a roadmap for *research* rather than an roadmap on industrial R & D in general. The roadmap is **not** a roadmap on real-time control. In real-time control the real-time computing system is used as an implementation platform for a control system controlling some external dynamical system, often a physical plant with external inputs and outputs. Here, it is instead the real-time computing system that is the subject to the control. The item that is controlled is in most cases the allocation of computing and communication resources, e.g., the distribution or scheduling of CPU time among different competing tasks, jobs, requests, or transactions. Due to this, control of computing systems also goes under the name of *feedback scheduling*.

The roadmap assumes basic knowledge in real-time computing and control engineering from the readers. In parallel to the current roadmap a separate roadmap on Real-Time Techniques in Control System Implementation has been developed.

1.2 Background

Feedback-based approaches have always been used in engineering systems. One example is the flow and congestion control mechanisms in the TCP transport protocol. Typical of many applications of this type is that feedback control is used in a more or less *ad hoc* way without any connections to control theory. During the last 5-10 years this situation has changed. Today control theory is beginning to be applied to real-time computing system in a more structured way. Dynamic models are used to describe how the performance or quality of service depend on the resources at hand. The models are then analyzed to determine the fundamental performance limitations of the system. Based on the model and the specifications control design is performed. In some cases the analysis and design is based on optimization.

The research on control of computing systems has increased immensely and gained a large interest during the last years. A large number of applications have been proposed in different areas, e.g., high-performance web servers [Diao *et al.*, 2002; Gandhi *et al.*, 2001; Lu *et al.*, 2001; Robertsson *et al.*, 2003], multimedia streaming [Nahrstedt, 1995; Zhang *et al.*, 2001], real-time databases [Amirijoo *et al.*, 2003d], web storage systems [Lu *et al.*, 2002], network routers [Keshav, 1993; Christin *et al.*, 2002], active queue management schemes [Hollot *et al.*,

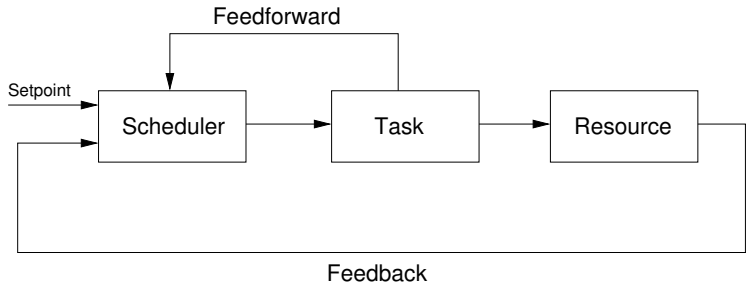

Figure 1 A feedback scheduler structure. Feedforward is used to adjust to known changes in required resources.

2001a; Hollot *et al.*, 2001b], processor architectures [Skadron *et al.*, 2001], and control systems [Cervin *et al.*, 2002]. Several of these applications will be described in further detail in the sequel. There is also a new textbook available [Hellerstein *et al.*, 2004]. However, so far most of the work presented in literature have been conducted by scientists working *either* in the real-time computing or telecommunication fields, *or* in the automatic control field. Unfortunately, this has sometimes led to erroneous models and strange results. One example is that most control-theoretic models of queuing systems have been linear and deterministic, rather than nonlinear and stochastic which are better descriptions of the true behaviour of a queuing system. Another example is incorrect assumptions made by control researchers due to lack of deep knowledge of the details of the computing and communication system.

1.3 Feedback scheduling

In a real-time system with hard timing constraints, e.g., deadlines, it is paramount that all timing constraints are fulfilled. If sufficient information is available about worst-case resource requirements, e.g., worst-case execution times (WCET), then the results from classical schedulability theory can be applied to decide if this is the case or not. Using, e.g., priority-based or deadline-based scheduling strategies, it is then possible to provide a system implementation that guarantees that the timing constraints are fulfilled at all times.

However, in many situations the hard real-time scheduling approach is unpractical. Worst-case numbers are notoriously difficult to derive. In order to be on the safe side a heuristically chosen safety margin is often added to measurements of "worst-case" values. This may lead to under-utilization of resources. In other cases resource requirements vary greatly over time. The reason for this may be changes in the external load on the system, e.g., large variations in the number of requests to a web-server, or mode changes in application tasks. Again, designing the system for the worst case may lead to under-utilization. The above situations are both caused by uncertainty. A major strength of control theory is its ability to manage uncertainty.

In feedback scheduling the allocation of resources is based on a comparison of the actual resource consumption by, e.g., a set of tasks, with the desired resource consumption (the setpoint value or the reference value). The difference, or control error, is then used for deciding how the resources should be allocated to the different users. The decision mechanism constitutes the actual controller in the feedback scheduling scheme. The structure of a feedback-based resource allocation scheme is shown in Figure 1. In the figure we assume that the resource

consumers are tasks that need a certain amount of CPU time each. The setpoint of the controller/scheduler is the desired total CPU utilization. If a task knows beforehand that it is about to change its resource consumption, it may inform the scheduler about this directly. This constitutes a feedforward path in the controller. Another name for a feedback loop is a *closed loop*. In relation to this a conventional scheduling algorithms can be described as operating in *open loop*, without any mechanisms that allow it to adjust to changes in loads, overruns, etc.

A key observation here is that feedback scheduling is not suitable for applications that are truly hard in nature. The reason for this is that feedback acts on errors. In the CPU utilization case above this would mean that some tasks temporarily might receive less resources than required, i.e., they could miss deadlines. Feedback scheduling is therefore primarily suited for applications that are *soft*, i.e., tolerate occasional deadline misses without any catastrophic effects, or that are said to be *adaptive*. The latter means that missing one or more deadline does not jeopardize correct system behavior, but only causes a performance degradation. For this type of systems, the goal is typically to meet some *Quality of Service* (QoS) requirements.

The adaptive class of real-time systems is a suitable description for a many practical applications. This includes different types of multimedia applications, and web server systems. It also includes a large class of control applications. Most control systems can tolerate occasional deadline misses. The control performance or *Quality of Control* (QoC) is also dependent on to which degree the timing requirements are fulfilled. It is only in safety-critical control applications, e.g., automotive X-by-wire applications, that the hard real-time model really is motivated.

1.4 Important issues

Important issues in control of computing systems are what the inputs and outputs of the systems are, the structure and type of controller, and which modeling formalism that is employed.

Inputs and outputs An important issue in all control problems is what the inputs and outputs are. The input to the controlled system, i.e., the output of the controller or the *control variable*, is the means by which the controller changes the resource allocation. The output of the controlled system, also called the *measured variable*, is the variable or signal that the controller aims to maintain under control, i.e., keep it at a constant desired *setpoint* or have it follow a changing setpoint. The *actuator* is the mechanism through which the control variable is entered into the controlled system. In a web-server control application the control variable could be the total amount of work caused by pending requests whereas the actuator could correspond to an admission control mechanism. The *sensor* is the mechanism that is used to actually measure the measured variable. In a CPU utilization control application the sensor could correspond to measurements of the tasks' actual execution time. In order to keep the controller from over-reacting to spurious upsets in the measured variable, i.e., occasional overruns, a low-pass filter is often included in the sensor.

Controller structure and type Another important issue is the structure and type of the controller. In a feedback structure the controller bases its actions on the measured variable and the setpoint only. In a feedforward structure the

actions are based only on the setpoint and/or on measurable disturbances acting on the controlled system. In a combined feedback and feedforward structure the feedforward path is typically used to provide a fast response to setpoint changes whereas the feedback path is used to compensate for errors caused by disturbances acting on the controlled system, or incorrect modeling assumptions. In a single-input single-output (SISO) structure the controller controls a single measured variable using one control variable, whereas in a multi-input multi-output (MIMO) structure several measured variables and control variables are used. A common controller structure is the cascade controller where two ordinary controller are connected in series, the control variable of the first, outer, controller being used as the setpoint of the second, inner controller.

The controller type decides how the control variable is calculated based on the measured variable and setpoint. A common controller type both in control of computer systems and in control in general is the PID controller. In the PID the control variable is formed as combination of three terms: a proportional term, an integral term, and a derivative term. In the proportional term the control variable at time k, $u(k)$, is proportional to the control error at time k, i.e.

$$u(k) = k_p(y_{ref}(k) - y(k)) = k_p e(k),$$

where $y(k)$ is the measured variable at time k, $y_{ref}(k)$ is the setpoint (or reference value) at time k, and k_p is the proportional gain. In the integral term the control variable is proportional to the integral of the control error, i.e.,

$$u(k) = u(k-1) + k_i e(k),$$

where k_i is the integral gain. The integrator is hence implemented through accumulation. Finally, in the derivative part the control variable is proportional to the derivative of the control error, i.e.,

$$u(k) = k_d(e(k) - e(k-1)),$$

where k_d is the derivative gain parameter. Other common controller types are controllers on input-output form and on state-space form. The order of the controller corresponds to the number of old variables, i.e., the state, that must be stored in order to calculate the the control variable. For example, a proportional controller is of zero order and a PI controller is of first order.

The above controller types are linear. In a nonlinear controller the control variable is a nonlinear function of the controller inputs. In an adaptive controller the controller parameters vary over time based on changing conditions, whereas in a non-adaptive controller the controller parameters are constant. Hence, the meaning of the word *adaptive* is quite different in the computing community compared to the control community. In the computing community an ordinary controller with constant parameters, i.e., a nonadaptive controller from the control point of view, is often considered as adaptive, since it generates different control signals for different external conditions, i.e., it adapts it behaviour to the external conditions. For example, the name adaptive resource management is used in the computing community to denote resource management systems where the resources allocation is changed dynamically based on resource requirements and availability. From a control point of view a more adequate name for this would be dynamic resource management or controlled resource management.

For a general background on computer-based control, see [Åström and Wittenmark, 1997], on PID control, see [Åström and Hägglund, 1995], on adaptive control, see [Åström and Wittenmark, 1995], and on control of computing systems, see [Hellerstein *et al.*, 2004].

Modeling Formalisms When designing a controller two ways can be followed. In the heuristic approach a controller structure is selected based on experience and heuristics and the controller parameters are tuned manually. Although this approach works well in many cases, in particular for low-order controllers with few parameters, the approach has no theoretical foundation. In the model-based approach a model of the controlled systems is developed and this model is then used during the design of the controller. The model describes the dynamic relationship between system inputs and outputs. Due to the amazing properties of feedback it is often sufficient with a quite coarse-grained model that only captures the dominating dynamics and still achieve satisfactory performance.

When a computer-based controller is controlling a physical system (often denoted *plant*) sampling is employed. The, normally continuous, outputs of the plant are sampled with a certain sampling interval. This transforms the continuous-time signals to discrete-time series which are then used by the controller to generate the control variables, which also are discrete-time series. The digital-analog converter often works as a zero-order hold device generating a piecewise constant output signal which then is fed to the plant via the actuator. When sampling continuous-time signals and systems the sampling period must be chosen with care. A too long sampling interval may result in poor performance or instability. Aliasing effects may introduce artificial disturbance frequencies into the system unless proper analog anti-aliasing filtering is used. Measurement noise which may be unavoidable also generate fundamental limitations on the performance that the controller can generate.

When controlling a real-time computing system several things are different. The controlled system is of a discrete nature and all variables are discrete. Sampling and hold operations are not necessary. Measurement noise caused by the environment is not such a large problem any more.

Although several tings become easier control of computing systems also introduce new problems. A main problem is the lack of first principles models. When controlling a physical plant the laws of nature decide to a large degree the behaviour of the plant and can be use to derive dynamical models. Some examples are mass balances, energy balances, and momentum balances. A computing system, on the other hand, is a man-made artifact whose internal behaviour is not governed by any laws of nature, at least not on the macroscopic level. This means that it is, generally, not possible to derive any first principles models. One exception, where theoretical models are available, though, is queuing theory [Kleinrock, 1975; Robertazzi, 1994]. Queuing theory models have also been used with some success in the design of computing-system controllers. A drawback with queuing models is that they in most cases only hold in the average case.

Computing systems are discrete-event dynamic systems (DEDS), [Cassandras and Lafortune, 1999]. This and the fact that they are real-time systems makes it natural to use a timed discrete-event formalism, such as timed automata or timed Petri nets for modeling these systems. A drawback with this approach is that it is in many cases too fine-grained and easily leads to state-space explosion.

This typically is the case in queuing control systems when the arrival and departure rates are large. Another issue is the types of problems that these formalisms typically lend themselves to. Automata-based formalisms are well-suited for expressing and analyzing safety properties and blocking properties. Safety properties are concerned with the reachability of certain undesirable states, which could model undesirable or faulty conditions. Blocking properties are concerned with issues like deadlock and livelock.

Safety and blocking problems are, however, not the main concerns in performance control of real-time computing systems. Instead, it is issues such as stability, performance, and robustness that are prime concerns. For these types of problems a time-driven approach is more natural. However, the lack of first principles knowledge necessitates a system identification-based approach, in which a discrete-time model, typically a difference equation, is derived from measured input and output data. One example of this the fitting of a discrete-time model to measurement data using a least-square approach. The models derived in this way are based on periodic sampling. Likewise, the controllers designed from this type of models are based on periodic sampling. Although periodic controllers are possible in real-time computing, and are also, to a large extent, the approach that is mostly used in applications, it is from many respects more natural to invoke the controller in an event-driven fashion. For example, in a queue length control problem it makes more sense to calculate a new control action when a request is queued or dequeued, or every n'th queue/dequeue event, rather than periodically. An event-based controller would also be better conditioned for controlling the transient behaviour of the system, rather than the average behaviour.

A problem with aperiodic or event-based systems and aperiodic control of this type, rather than the DEDS type, is the lack of theory. This is because the resulting system descriptions are both time-varying and non-linear and hence very difficult to analyze. However, there are several indications from the field of control of physical systems that event-based control can have substantial advantages.

Event-based control of first-order stochastic systems was studied in [Åström and Bernhardsson, 1999]. It was shown that an event-based controller for an integrator plant disturbed by white noise requires, on average, only one fifth of the sampling rate of an ordinary, periodic controller to achieve the same output variance. Similarly, in [Speranzon and Johansson, 2003] event-triggered vs. periodic communication between mobile robots in a pursuit-evasion game was studied. Monte Carlo simulations showed that the event-triggered approach only required on average half the bandwidth of the periodic case.

1.5 Control of queues

Many aspects of the real-time performance of computing systems can be inferred from the behaviour of resource queues. Some examples of such queues are ready queues, semaphore queues, communication socket queues, and web request queues. A queue can be modelled in various way. Using queuing theory several types of models can be developed. One example is the so called Tipper's nonlinear flow model [Tipper and Sundareshan, 1990]. The model consists of a nonlinear differential equation for the steady-state behavior of the queue length of a M/M/1 or M/G/1-queue. As shown in [Robertsson *et al.*, 2003] this model can be used for analysis and control design of different types of controllers. At a high level, a queue can be seen as an integrator of request flows. This can be modeled using, e.g., a difference equation and then analyzed with control theory.

Flow models of queuing systems approximate the steady-state behaviour of the queue and, are typically more accurate the higher the load is on server. However, for small loads when the queue in most cases is empty, or, if the queue has a limited number of entries, when the queue is nearly full, these types of models are less appropriate. An alternative, then, is to model the queue as a discrete event system using, e.g., automata or timed automata.

In a queue there it is the difference between the service rate and the arrival rate that determines the delay experienced by the requests. Two types of actuators can be used. An *enqueue* actuator influence the arrival rate of the queue. An example of this is an *admission control* mechanism. Another example would be to change the inter-arrival period for periodic requests, e.g., by changing the period of a periodic task creating jobs that added to the ready queue. A *dequeue* actuator instead influences the service rate of the requests. Examples of this type of actuator mechanisms are different forms of quality adaptation. By reducing the quality provided, the CPU resources needed are reduced and, hence, the service time for the request.

An open question in queue length control is how to combine queuing models with control-theoretic methods. A common approach in delay control is to use nonlinear models from queuing theory for feedforward combined with simple feedback control of PID type. The aim of the feedforward path is to provide fast setpoint responses, whereas the role of the feedback controller is to compensate for disturbances and incorrect modeling assumptions. An example of the latter is incorrect assumptions about the stochastic nature of arrival and departure processes. A common assumption is that these are Poisson processes, something which is far from true for typical web traffic.

2. Current Industrial Practice and Needs

Control of computer system is still in its infancy with respect to industrial development and applications. The work on feedback scheduling in real-time operating systems is still only at a research stage. It is not until basic support for this is available in commercial kernels and OS that the area has the potential to really grow. The same situation holds for control of server systems. However, the real-time computing industry is quite conservative, in most cases still only supporting basic priority-based scheduling. Still, partition-based temporal isolation between tasks is available in some commercial OS, e.g., Integrity from Green Hills Software. An important way to get industrial acceptance for this research is to create standards. One example would be a control extension to POSIX, i.e., POSIX/Control.

In 2001 IBM started the autonomic computing initiative (ACI), [Kephart and Chess, 2003]. The ultimate aim is to create self-managing computer systems to overcome their rapidly growing complexity and to enable their further growth. ACI encompasses the following four functional areas:

- Self-Configuration: Automatic configuration of components;
- Self-Healing: Automatic discovery, and correction of faults;
- Self-Optimization: Automatic monitoring and control of resources to ensure the optimal functioning with respect to the defined requirements;

- Self-Protection: Proactive identification and protection from arbitrary attacks.

One of the bases for autonomic computing is closed loop control. The activities of IBM within control of, e.g., server and data storage systems, are part of the autonomic initiative. The autonomic computing ideas have also been adopted by other companies, e.g., Hitachi, [Iijima *et al.*, 2002]. Large end users of web server technology such as Amazon and Google also have a considerable interest in control-based approaches to performance control. For example, Amazon apply feedback control in their servers already today and has its own internal development group within the area. A workshop on the future trends in control of computer systems that was organized by NSF in May 2005 had industrial participants from IBM, Microsoft, HP, and Amazon.

Control-based or adaptive resource management is a research area of strong interest in particular to companies within the multimedia market, e.g., Philips. However, adaptive resource management is a promising technology for all companies that apply COTS technology in applications with resource limitations. One example of this is industrial automation.

3. Control of Servers

More and more business and services rely on Internet technology and server technology. Control has a natural position here, both within the communication network and at the server side. Queue management is important in all servers, e.g., web servers. Requests to the server are stored in an input queue, the server or worker thread servicing the requests are stored in the ready queue or in different waiting queues, e.g., in order to access memory. The architecture is shown in Fig. 2.

3.1 Control-based queue admission control

Since queuing systems have a stochastic behavior it is difficult to find equations that are simple enough to use in the analysis. A nonlinear flow model that is well-suited for a control-based approach is the so called Tipper's model. It was first developed by [Agnew, 1976] and was further investigated in [Pitsillides *et al.*, 1995], [Sharma and Tipper, 1993], [Tipper and Sundareshan, 1990], and [Wang *et al.*, 1996]. In the references they show that the steady-state behavior of the queue length, x, of a single server queue is given by

$$\dot{x} = \lambda - \mu G(x(t)) \tag{1}$$

where λ is the mean rate of the arrival process and μ is the mean service rate.

For an M/M/1 system we have

$$G(x(t)) = \frac{x(t)}{x(t)+1} \tag{2}$$

and for an M/G/1 system the expression becomes

$$G(x(t)) = \frac{x(t)+1-\sqrt{x^2(t)+2C^2x(t)+1}}{1-C^2} \tag{3}$$

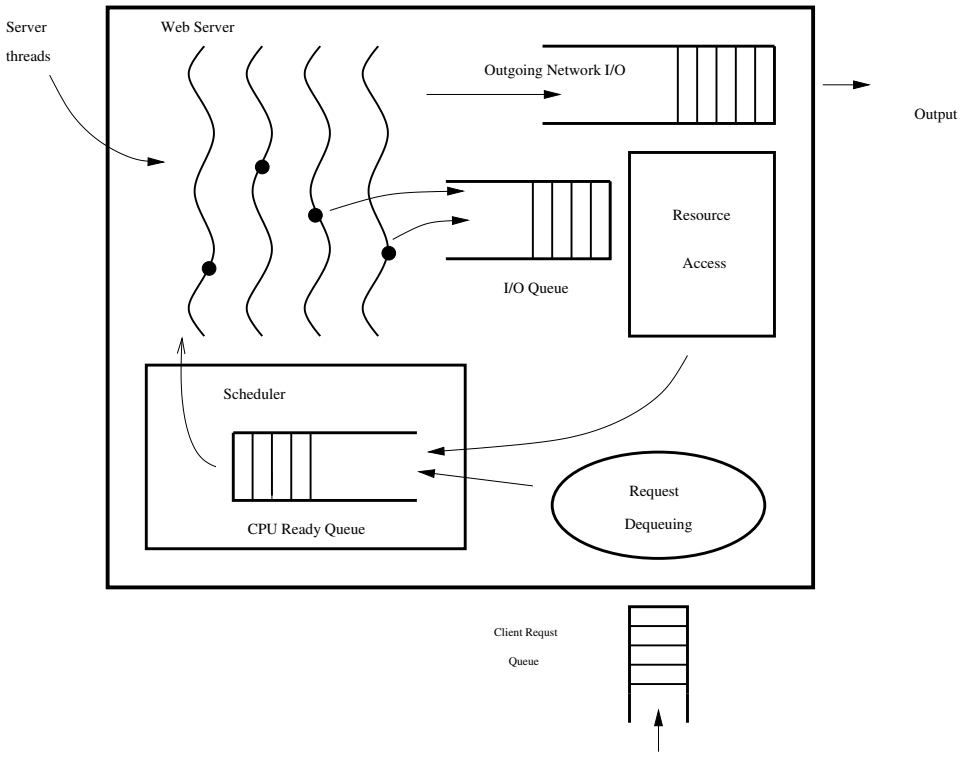

Figure 2 Web server

where C^2 is the squared coefficient of the variance of the service time distribution. It has been shown in the cited references that the model is correct in terms of average number of customers and service utilization during steady state.

Here we will use the M/M/1 version of this model to derive a admission controller based on PI-control. The first step is to introduce a control signal. The output of the controller, u, is the desired admission rate. This is used by a gate that rejects those requests that cannot be admitted using percent blocking. Hence, u is limited between 0 and 1. The model with the control signal is given by

$$\dot{x} = \lambda u - \mu \frac{x(t)}{x(t)+1}. \tag{4}$$

The next step is to linearize this equation around a certain operating point $x = x^\circ$. If we let $y = x - x^\circ$ the linearized model becomes

$$\dot{y} = \lambda y - \mu \frac{1}{(x^\circ+1)^2} y = \lambda u - \mu a y \tag{5}$$

with $a = 1/(x^\circ+1)^2$. We now control this model with a PI-controller given by

$$u(t) = K(e(t) + \frac{1}{T_I}\int e(\tau)d\tau). \tag{6}$$

The closed loop system on Laplace transform form given by

$$G_{cl}(s) = \frac{\lambda K(s+\frac{1}{T_I})}{s(s+\mu a) + \lambda K(s+\frac{1}{T_I})} \tag{7}$$

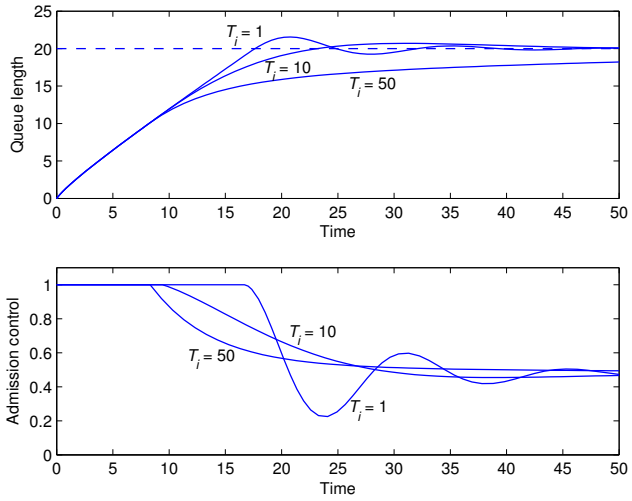

Figure 3 Tipper's nonlinear flow model controlled using a PI-based admission controller.

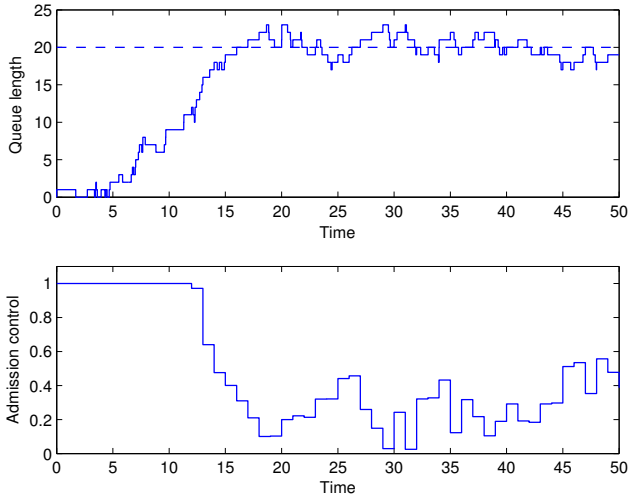

Figure 4 An M/M/1 queue controlled using a PI-based admission controller.

By choosing K and T_I we have full freedom to position the poles of the closed loop system, i.e. get the desired speed and damping. Simulations using the PI controller on the nonlinear queuing model with $\lambda = 2$, $\mu = 1$, $x^\circ = 20$, $K = 0.1$ and different values of T_i are shown in Figure 3. Here, the PI-controller has been extended with an anti-windup mechanism to prevent the integrator state to grow out of bounds during the initial transient when the control signal is limited to 1, i.e. all requests are admitted. Anti-windup is essential in all controllers with integral action to achieve good performance during transients.

Finally, in Figure 4 the PI controller is applied to a simulation of a M/M/1 model with discrete arrivals and departures. Here, the PI-controller is implemented in discrete-time. As can be seen the controller behaves nicely also on the real process. In [Robertsson *et al.*, 2003] the stability of the above controller applied to the nonlinear flow model in the M/G/1 case is analyzed and proved. The same nonlinear flow model can also be used in an analogous way, if we instead control the service times of the individual requests, i.e., if we use the service rate μ as the control signal.

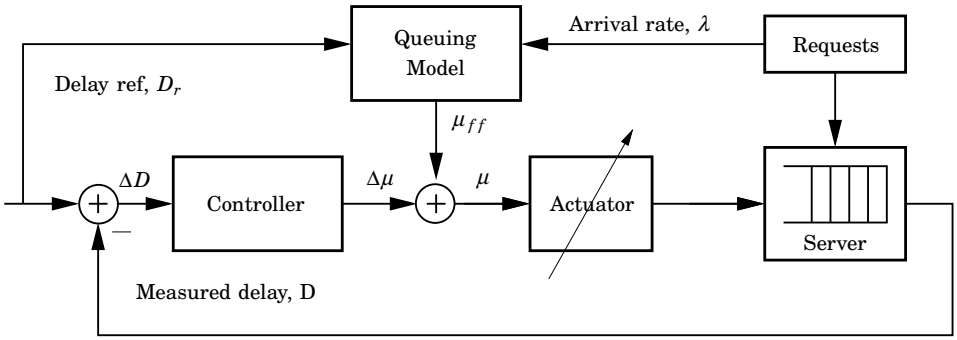

Figure 5 A feedforward + feedback service rate controller.

3.2 Control-based queue delay control

The delay that a server request experiences can be effected in different ways. One possibility is to increase the number of server, or worker, threads. Another possibility is to use quality adaptation or to modify the processor speed using, e.g., dynamic voltage scaling. In [Sha *et al.*, 2002] it was suggested to combine feedback control with queuing theory-based feedforward control. The aim was to keep the average timing delay experienced by the requests close to a desired value D_r. For an M/M/1 queue with with arrival rate λ and service rate μ, the long-term average queuing time for the requests is

$$D = \frac{1}{\mu - \lambda}. \tag{8}$$

The above equation can be solved for the value of μ that gives the desired delay D_r. This gives an equation for the feedforward term, i.e.,

$$\mu_{ff} = \frac{1}{D_r} + \lambda. \tag{9}$$

The feedback controller was a linear P or PI controller. The role of the feedback is to suppress incorrect modeling assumptions and transient errors around the operating point. The actuator of the control systems was the number of server threads reserved for the particular request class. The structure of the controller is shown in Figure 5.

Queuing-theoretic models describe relations between long term averages only. In the shorter time horizon, delay and rates may fluctuate. Internet traffic, for example, is generally very bursty and changes abruptly. A way to handle this is to sample quite seldom and to use the average value of the measured variable over the time window since the previous sample as the input to the controller. The drawback with this approach is that is less suited for handling transient situations. In [Amirijoo *et al.*, 2005] an approach is proposed that aims to overcome this problem. A suitably chosen sampling period that captures the system dynamics is combined with an estimator that produces estimations of the controlled variable.

Another drawback with queuing theory-based models is that they make certain restrictive assumptions about the arrival and service processes of the system, which are often poorly matched by reality. In real server queues, the statistic nature of the traffic may show considerable variations, and standard Poisson

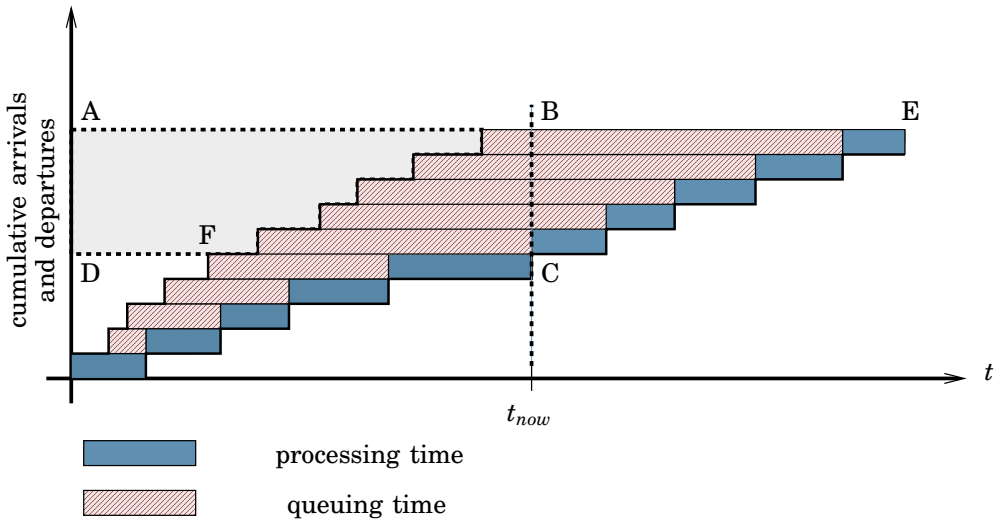

Figure 6 Server queuing and processing delay over time.

processes do not capture this behavior. Instead, the Pareto distribution has been reported to fit measurements of real web traffic well [Crovella and Bestavros, 1997]. This distribution has typically a long tail, and shows self-similar and long range dependent characteristics.

In [Henriksson *et al.*, 2004] an improved feedforward scheme is presented, that makes no assumptions about the statistical properties of the traffic. Instead, it predicts future delays as a function of instantaneous measurements of the situation in the server queue. This includes current queue length and the arrival times of the queued requests, which are assumed to be recorded for use in the prediction.

Figure 6 shows a geometric picture used to derive the predictor equation. The horizontal axis shows the evolution of time, and the vertical axis shows the cumulative number of arrivals and departures of requests. Each horizontal two-coloured block in the figure represents one request and is divided in queuing time and processing time. The vertical distance in the shaded area at any point in time (for example the distance CB at time t_{now}) represents the actual queue length.

For the situation in the figure it can be noted that the line from the origin to point B gives the average arrival rate of the ten first requests. Similarly, the line between the origin and point C represents the average service rate of these requests up until time t_{now}. Since the arrival rate exceeds the service rate, it can be seen that the delays experienced by the requests build up.

The basic idea with the predictor is to choose the service rate that achieves a desired average delay of the requests in the system taking into account their average queuing delay up until t_{now}. Geometrically this means modifying the slope of the line CE to obtain a desired area of the shaded area $CEBF$. By continuously updating the predictor as requests enter and leave the queue, sudden variations are taken care of more rapidly than using the queuing-theoretic models.

The results presented in [Henriksson *et al.*, 2004] are concerned with absolute delay control by manipulation of the service rate of incoming requests. Note, however, that the predictor could easily be used also in a scheme to adjust the

arrival rate using admission control. In that case one would change the slope of the arrival rate instead, under the assumption that the service rate is being constant.

The feedback controller used in [Henriksson *et al.*, 2004] is a gain-scheduled PI-controller based on feedback from actual delay measurements. Different controller parameters are used based on the current setpoint and the estimated arrival time. An observation window, N_{obs}, is used to accurately estimate the average values of arrival rates and precessing times of requests. The PI-controller is invoked in an event-triggered fashion, every n'th event (request departure), where $n < N_{obs}$. Also here, anti-windup was crucial to obtain good performance.

Simulations performed with TrueTime [Cervin *et al.*, 2003] showed that the instantaneous feedforward controller gave better results than the queuing theory-based feedforward. The same results were achieved using experiments on an Apache web server test-bed with the requests generated with the Scalable URL Reference Generator (SURGE).

An advantage with this approach compared to the approach based on a feedforward term based on queueing theory is that no assumptions on the statistics of request arrivals and departures are made. A short-coming of the scheme is the fact that it is the actual processing times of future requests are not known, but are estimated based on past measurements. However, every time a request departs the estimation is redone. This can be compared to the receding horizon principle employed in model-based predictive control (MPC).

3.3 Relative Queue Delay Control

A FIFO queue can naturally be modeled as an integrator on differential or difference equation form. The same does unfortunately not hold for a priority queue. Priority queues are common in real-time systems, e.g., the ready queue and semaphore queues in a real-time kernel or server request queues in the case the requests are divided into different service levels or classes represented by different priorities. The usual way of handling this situation when the number of priorities is small is to model the priority queue as a number of FIFO queues, one for each priority level.

A commonly proposed performance optimization criterion for multi-class systems is the weighted fairness guarantee. Here, the desired ratio of some performance metrics across different classes is specified rather than absolute values for each class. This is expressed as

$$\frac{D_{i+1}(k)}{D_i(k)} = \frac{c_{i+1}}{c_i}, i = 1, \cdots, N-1,$$

where $D_i(k)$ is the average performance (e.g., delay) of traffic class i at time k, and c_i is a constant per-class weighting factor representing the user's relative performance specifications. Several approaches have been proposed for multi-class ratio control. For example, in [Lu *et al.*, 2001; Lu *et al.*, 2002] a solution uses per-class control loops, where error $e_i(k)$ of $class_i$ at sampling time k is given by the expression $e_i(k) = \frac{c_i}{\sum_{j=1}^{N} c_j} - \frac{D_i(k)}{\sum_{j=1}^{N} D_j(k)}$. The controller computes a corresponding correction Δb_i to the current resource allocation b_i. In [Lu *et al.*, 2003], the average of $\frac{D_1(k)}{c_1}$, $\frac{D_2(k)}{c_2}$, ..., $\frac{D_N(k)}{c_N}$ is taken as the common set point

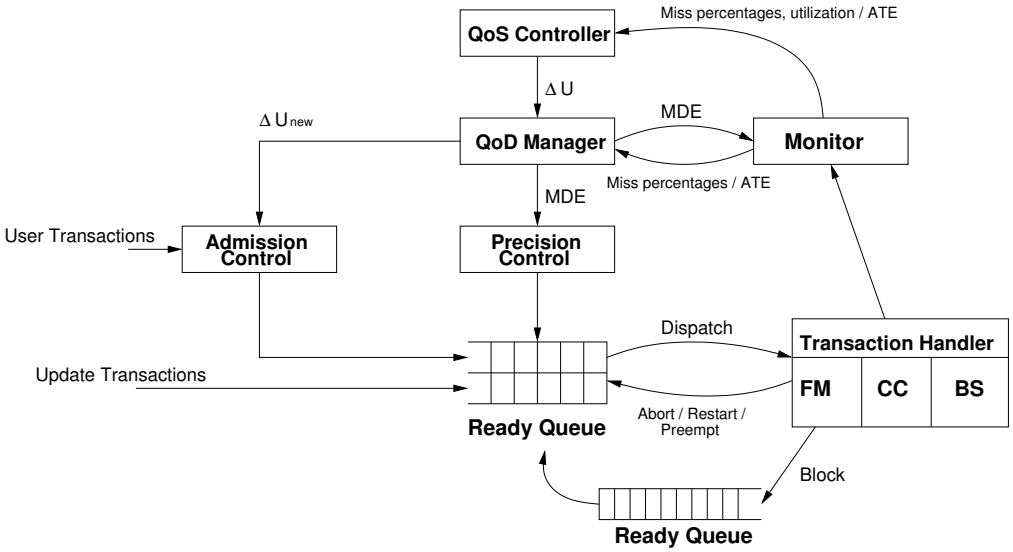

Figure 7 Database QoS control architecture from [Amirijoo *et al.*, 2003c]

letting individual ratios converge to their average. In [Lu *et al.*, 2001], the performance error of consecutive class pairs is defined as $e_i(k) = \frac{c_{i+1}}{c_i} - \frac{D_{i+1}(k)}{D_i(k)}$. It is not clear which of these approaches that generally is the best. Also the relationship between this type of ratio control and the type of ratio control employed in process control is unclear.

3.4 Control of Real-Time Databases

Real-time database servers is another area where control techniques have been proposed. These types of systems are being increasingly used within several areas, e.g., manufacturing, telecommunication systems and eCommerce. The desire for vertical integration in, e.g., manufacturing systems, results in databases containing data of highly different type, ranging from real-time sensor data to high-level management and enterprise data.

Real-time databases must maintain both the logical consistency of the database (integrity constraints) and its temporal consistency (meeting the deadline of the transactions). Due to the difficulties in predicting the workload of real-time databases, transaction deadline misses and data freshness violations may occur during transient overloads. To overcome this problem feedback-based quality of service (QoS) approaches have been proposed.

In [Amirijoo *et al.*, 2003c] a common framework and architecture is proposed that that unifies earlier proposed algorithms FCS-IC-1, FCS-IC-2, FCS-HEF, and FCS-HEDF [Amirijoo *et al.*, 2003a; Amirijoo *et al.*, 2003b]. A main memory database model is used with a single CPU. Data objects are either temporal or non-temporal. A temporal data object d_i is is considered temporally inconsistent or stale if the current time is later than the timestamp of the data plus the length of the validity interval associated with d_i. Transactions are classified as update transactions or user transactions. Update transactions arrive periodically and update temporal data objects. User transactions arrive aperiodically and may read temporal data and read/write non-temporal data. Transactions are modeled as imprecise computations consisting of one mandatory subtransaction and a number of optional subtransactions. The mandatory subtransaction is necessary

for an acceptable result whereas the optional subtransactions are executed in sequence if there is enough time and resources available. Update transactions are assumed to only have a single mandatory subtransaction. Quality of service specifications are used both for data (quality of data (QoD)) and for transactions (quality of transactions (QoT)). Each performance metric is specified in terms of its reference value to express the stationary desired value and its overshoot and settling time to express the transient performance. QoD is expressed in terms of MDE (Maximum Data Error) which is the maximum value of the relative error of a temporal data item. The QoT is specified either as the deadline miss percentage of optional subtransactions or in terms of the average transaction error (ATE) which measures the percentage of user transactions that are able to complete all their subtransactions.

The architecture is shown in Fig.7. Admitted transactions are placed in the ready queue. The transaction handler manages the execution of the transactions. When the QoT specifications is defined as the deadline miss percentage of optional subtransactions the deadline miss percentage of mandatory user subtransactions and of optional user subtractions are used as the controlled variable, whereas in the other case the ATE is used as the controller variable. At each sampling instant the controlled variables are monitored and fed to the QoS controller which calculates a change, ΔU, to the total estimated requested utilization. Based on ΔU the QoD manager changes the total estimated requested utilization by modifying the QoD (i.e., adjusting MDE). The precision controller then schedules the update transactions based on MDE. The part of ΔU that cannot be handled through the QoD adjustments is taken care of by the admission controller. The transaction handler consists of a freshness manager (FM), a concurrency control unit (CC), and a basic scheduler (BS).

The different control algorithms use different combination of controllers, e.g., FCS-IC-1 uses one utilization controller and two miss percentage controllers (one for the miss percentage of optional subtransactions and one for the deadline miss percentage of mandatory user subtransactions)., whereas FCS-IC-2 uses two miss percentage controllers. Both algorithms uses EDF as the basic scheduler. The controllers are nonlinear algorithms, which are tailored to the specific problem at hand rather than being based on, e.g., PID type of ideas. The FCS-HE and FCS-HEDF control QoT through ATE. They are also designed to enhance QoS fairness among transactions.

3.5 Challenges and Research Directions

The main objective in control of servers and software systems in general is to derive a unified theory and framework for performance control of queuing system that combines elements from control theory and queuing theory and which allows an integration of both time-driven liquid model formalisms and event-driven formalisms.

Several major challenges exists:

- **Modelling challenges:** Which is the right or optimal abstraction level for this type of control problem is still a question with no clear answer. Models at different levels and types need to be combined. In some cases purely static models can be sufficient. For high load situations differential equation-based liquid flow models can be applied. However, how to handle medium-load traffic situations and rapid and large traffic changes is still

a challenge. It is also sometimes desirable to combine time-based models with event-based discrete models. Hence, we need better understanding for which models types that are best suited for a particular applications. It is also possible that new models types must be derived for this type of problems.

- **Control challenges:** The challenges for control are connected to the modelling challenges above. How do we develop a control theory based on this type of models. In certain cases symbolic values are better suited than numeric values to describe the performance of software systems. For example specifications and setpoints may be expressed in natural language of using logical predicates rather than as real numbers. Control based on symbolic representations is an active research area which so far not has been applied to this type of problems. In control of server systems where requests arrive and depart in a event-based fashion, it is natural to implement a controller in a event-based way, i.e., invoke the controller when a request departs or at every n'th arrival, also if the controller has been designed in a time-driven fashion assuming periodic sampling. This combination creates several challenges.
- **Software dynamics:** Our current notion of dynamics is based on the behaviour of physical systems, e.g., mechanical systems. It is not necessarily so that this type of dynamics also are suitable for software systems. The same holds for stability. It is not completely clear what an unstable software system really means or what type of stability definitions that make sense. Related to this is the question of how we design or program software systems in such a way that they are observable and controllable. Which types of sensor and actuators makes most sense for this type of systems.

In addition to research motivated by the objective and challenges above the following research directions are important:

- **Control Patterns:** In control in general and in particular in process control the characteristics of different types of control loops and control problems are well known and even in some cases formally categorized. Examples of different types of control loops with different characteristics are flow control, temperature control, pressure control, and composition control. Similarly a number of well-defined controller structures exist, e.g., cascade control, ratio control, min-max selector structures etc. The same type of classification is necessary also in control of computer systems. One possibility is to make use of ideas from design patterns to create well-defined patterns of control for server control systems. For instance, in which situation should a queue control problem be formulated as an admission control problem and when should it be formulated as a delay control problem.
- **Operating System Support:** In order to make control of server systems applicable on a wider industrial scale it is necessary to have built-in support for this is operating systems and/or middleware. On which level this should be is not clear. Should there be a POSIX/Control standard with built-in support for feedback control?
- **Control of the Software Engineering Process:** The largest bottleneck in the production of software-intensive systems today is the development

process itself, including requirements engineering and testing. Tight feedback with frequent incremental tests is a vital part of most agile development process, e.g., eXtreme Programming (XP). A large challenge is to apply control engineering methods in a formal way not only to the performance control of the software systems but also to the software engineering process itself.

- **Control of multi-tier systems:** Large eCommerce servers are multi-tier systems consisting of web server front-ends, business logic in the intermediate layers, and database servers as back-ends. A client request to the overall server propagates through the different tiers and give rise to sub-requests. The overall system is a MIMO system where control is needed at several layers. Control for this type of systems is an important research direction. Model-based predictive control (MPC) is one interesting possibility.
- **Handling of constraints:** Constraints on control signals and state variables are common in queue problems and server problems. A queue can never contain less than zero entries and memory constraints limit the maximum number of entries. In admission control it is only possible to admit $0 - 100\%$ of all the requests. Methods for handling constraints are therefore important in this type of control problems. One example is anti-reset windup techniques to avoid problems with control signal limitation in the presence of integral action in the controller. Again, the possibility to include constraints in the control problem formulation makes MPC an interesting candidate technology.
- **Performance metrics:** In software systems additional metrics are important in addition to performance. These include security, reliability, availability, efficiency, etc. Is it possible to also include these in the control problem formulation? This issue strongly relates to error control of software which is discussed in Section 8.

4. Control of CPU Resources

Feedback scheduling of CPU resources is an area where fairly much research has been performed. In feedback scheduling the allocation of CPU resources is based on a comparison of the actual resource consumption by, e.g., a set of tasks, with the desired resource consumption (the setpoint value or the reference value). The difference, or control error, is then used for deciding how the resources should be allocated to the different users. Feedback scheduling is primarily suited for applications that with soft or adaptive real-time requirements. This includes different types of multimedia applications, but also a large class of control applications. Most control systems can tolerate occasional deadline misses. The control performance or *Quality of Control* (QoC) is also dependent on to which degree the timing requirements are fulfilled.

Feedback scheduling pf CPU resources has strong relationships with the queue control employed in server systems. Controlling the delay experienced by task jobs in the ready queue is not very much different from controlling the delay experienced by web requests. This is particularly true if we consider aperiodic tasks. Hence, many of the results in one area can be directly applied in the other area.

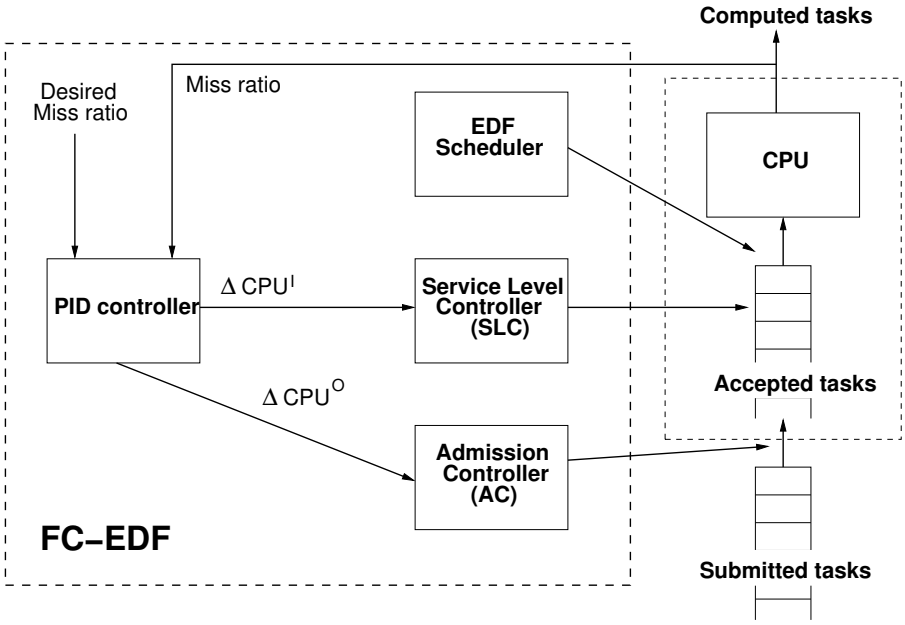

Figure 8 The EDF-FC scheme (from [Stankovic *et al.*, 1999])

The idea of using feedback in scheduling has to some extent been previously used in general purpose operating systems, in the form of multi-level feedback queue scheduling [Kleinrock, 1970; Blevins and Ramamoorthy, 1976; Potier *et al.*, 1976]. However, this has mostly been done in an ad-hoc way.

4.1 Feedback-Based Task Scheduling

A more control-theoretical approach to task scheduling is taken in [Stankovic *et al.*, 1999; Lu *et al.*, 1999] that present a scheduling algorithm called Feedback Control EDF (FC-EDF). A PID controller regulates the deadline miss-ratio for a set of soft real-time tasks with varying execution times, by adjusting their CPU utilization. It is assumed that tasks can change their CPU consumption by executing different versions of the same algorithm. An admission controller is used to accommodate larger changes in the workload. The scheme is shown in Fig. 8.

In [Lu *et al.*, 2000] the approach is extended. An additional PID controller is added that instead controls the CPU utilization. The two controllers are combined using a min-approach. The resulting hybrid controller scheme, named FC-EDF^2, gives good performance both during steady-state and under transient conditions. The framework is further generalized in [Lu *et al.*, 2002], where the feedback scheduler is broken down in three parts: the monitor that measures the miss ratio and/or the utilization, the control algorithm, and the QoS actuator that contains a QoS optimization algorithm to maximize the system value.

Many scheduling techniques that allow QoS adaptation have been developed. An interesting mechanism for workload adjustments is given in [Buttazzo *et al.*, 1998], where an elastic task model for periodic tasks is presented. The relative sensitivities of tasks to period rescaling are expressed in terms of elasticity coefficients. Each task is characterized by five parameters: computation time C_i, a nominal period T_{i_0}, a minimum period $T_{i_{min}}$, a maximum period $T_{i_{max}}$, and an elasticity coefficient $e_i \geq 0$. A task may change its period within its bounds.

When this happens the periods of the other tasks are adjusted so that the overall system is kept schedulable. An analogy with a linear spring is used, where the utilization of a task is viewed as the length of a spring that has a given rigidity coefficient ($1/e_i$) and length constraints. The elasticity coefficient is used to denote how easy or difficult it is to adjust the period of a given task (compress the string). A task with $e_i = 0$ can arbitrarily vary its period within its range, but it cannot be varied by the scheduler during load reconfiguration. The approach can be used under fixed or dynamic priority scheduling. Schedulability analysis of the system under EDF scheduling is given. In principal it is possible to modify the approach so that it also adjusts execution times. In [Gill *et al.*, 1998], a mixed static/dynamic-priority scheduling approach for avionics systems is presented. Each task is associated with a criticality parameter. In overload situations, tasks at the highest criticality level are allowed to execute before other tasks. Similar ideas are used within the broader area of *value-based scheduling*, e.g., [Burns *et al.*, 2000].

The End-to-end Utilization CONtrol (EUCON) algorithm, [Lu *et al.*, 2004], employs a distributed performance feedback loop that dynamically enforces desired CPU utilization bounds on multiple processors in distributed real-time embedded systems. EUCON is based on a model predictive control approach that models the utilization control problem on a distributed platform as a multi-variable constrained optimization problem. A multi-input-multi-output model predictive controller is designed and analyzed based on a difference equation model of distributed real-time systems.

4.2 Feedback-based Bandwidth and Reservation Allocation

Another area where control-based ideas have been employed is for dynamic allocation of bandwidth in aperiodic task servers and for dynamic allocation of resource reservations in reservation-based scheduling. Since aperiodic task servers of the constant bandwidth server (CBS) type, [Abeni and Buttazzo, 1998], are commonly used to implement reservation-based scheduling in systems with dynamic priorities (EDF), see [Abeni and Buttazzo, 2004], here the presentation will be focused on feedback-based control of resource reservations. The main application area for this techniques is multimedia applications, e.g., streamed audio and video.

The idea behind resource reservation is to explicitly control the computing resources assigned to a given activity (job, task, or application). Each activity receives a fraction (reservation) U_i of the processor capacity and will behave as if it was executing alone on slower, virtual processor. If an activity attempts to exceed its allocated reservation it will be delayed, preserving the resource for other activities. Through resource reservation the experienced QoS of a task will depend of how large reservation that has been reserved for the service. The main benefit of resource reservation compared to using task priorities to express relative importance is that it provides temporal isolation between tasks.

The motivation for feedback control in combination with resource reservation is the need to cope with incorrect reservations, to be able to reclaim unused resources and distribute them to more demanding tasks, and to be able to adjust to dynamic changes in resource requirements. Hence, a monitoring mechanism is needed to measure the actual demands and a feedback mechanism is needed to perform the reservation adaptation. Two types of feedback are possible. On a global, system-wide level a QoS controller adjusts the size of the individual

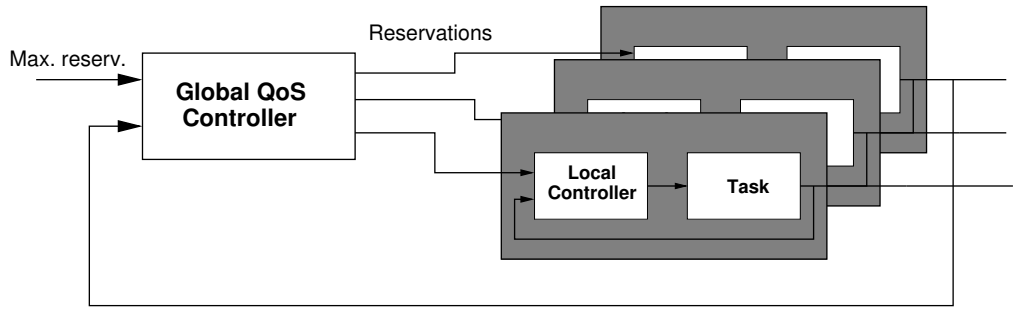

Figure 9 Hierarchical reservation control

reservations given to the different activities based on the measured performance and resource utilization. On a task or activity level local feedback is employed to adjust the resource requirements of the individual tasks based on the experienced QoS levels and the amount of resources available to the task, as decided by the global QoS controller. The local resource requirements can be done by rate adaptation, executing the task at different service levels using, e.g., imprecise computations or multiple version, and job skipping. The resulting feedback scheduling structure is hierarchical or cascaded and shown in Fig. 9.

A large amount of feedback-based or adaptive global QoS management systems have been proposed. Some examples are [Chu and Nahrstedt, 1999; Aparah, 1998; Nakajima, 1998]. These systems does typically not consider the local application behaviour. In [Abeni *et al.*, 2002], the problem of dynamically assigning bandwidths to a set of constant bandwidth servers is analyzed. A PI-based controller structure is suggested. In [Palopoli *et al.*, 2003] the authors propose an hybrid control approach. The servers are modeled as discrete switched systems, and a feedback scheduler that adjusts the server bandwidths is derived using hybrid control theory. Finally, in [Cucinotta *et al.*, 2004] they propose combining feedback based on a stochastic dead-beat controller with a feedforward moving average predictor. In [Abeni and Buttazzo, 2001] an adaptive reservation strategy is proposed for controlling the CPU bandwidth reserved to a task based on QoS requirements. A two-level feedback control is used to combine local application level mechanisms with global system-level strategies.

4.3 Challenges and Research Directions

Many of the challenges and research directions in control of CPU resources are similar to the ones for control of server systems. This includes in particular the modeling challenges. Some the more specific research directions are the following:

- **Multiprocessor systems:** Multiprocessor systems will become common in the near future also for certain embedded applications. So far very little of the control-based methods to CPU resource management have been applied to multiprocessor systems.
- **Power-aware CPU scheduling:** Adjusting the CPU speed using, e.g., Dynamic Voltage Scaling (DVS) techniques, is an alternative way of adjusting the service requirements of a task. Minimizing the power consumption is also an important goal in itself for many networked embedded systems, e.g., sensor networks. The joint optimization problem of minimizing energy

while still meeting real-time constraints already today receives a considerable attention from the research community. However, it is an important area also for the future.

- **End-to-end resource management:** Resource management in distributed systems where an activity spans multiple nodes is an important issue. How do we adapt the resources individually in the different nodes in order to obtain a good global behaviour, e.g., acceptable end-to-end response times?
- **Control-theoretic adaptive resource management:** Although adaptive resource management is a well-established area it is still not common that control-theoretic methods are being applied in the analysis and derivation of the adaptation mechanisms. Here there is room for more research.
- **Hierarchical resource allocation schemes:** Hierarchical resource allocation schemes based on dynamic reservations in combination with local feedback control loops for the individual tasks is an interesting and promising approach where more research is needed. How do we enforce the notion of virtual CPUs that execute within a real CPU with, possible, different scheduling policies, and where the share that each virtual CPU receives of the total CPU resources is dynamically adjusted based on resource requirements and availability?
- **Efficient feedback scheduling mechanisms:** One of the goals of feedback scheduling is to better make use of scarce resources. If this should be doable it requires that the feedback scheduling mechanism itself does not consume too much resources. Hence, efficient feedback scheduling mechanisms are of great importance. Methods that, e.g., require the on-line solution of an optimization problem at each invocation have very limited applicability.

5. Feedback Scheduling of Control Systems

Feedback-based resource scheduling is of particular interest for control systems. Here we assume that a control system involving multiple control loops is implemented as a multi-tasking system with each controller being realized as a separate periodic task. The main resource of concern in these types of problems is the CPU-time. The objective for the feedback scheduler is to dynamically adjust the CPU utilization of the controller tasks so that the task set remains schedulable and the stability and performance requirements of the individual controllers are met.

The structure is shown in Fig. 10. The controller are denoted $C_i(z)$ and the physical plants are denoted $P_i(s)$. Control is used at two levels: to control a number of physical plants and to control the resource allocation to the controllers.

In this approach the control performance can be viewed as a QoS parameter. The feedback scheduling problem is often stated as a optimization problem where the objective is to maximize the global control performance according to some criterion, subject to resource and schedulability constraints.

There are several reasons why feedback scheduling can be applied to control systems. One reason is the uncertainty associated with the WCET estimation.

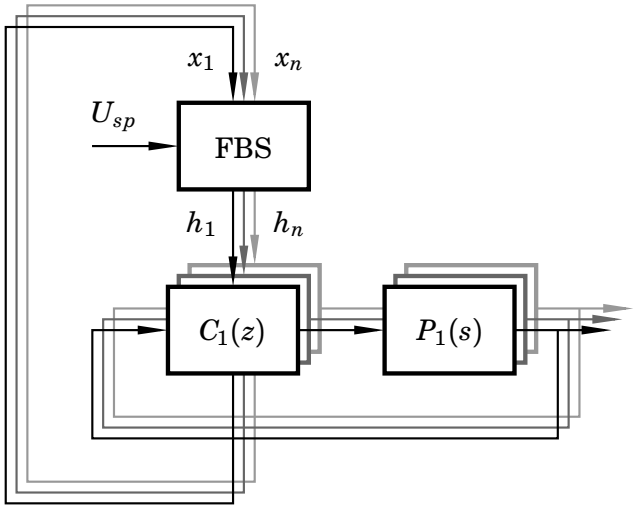

Figure 10 Feedback scheduling of control loops.

This is something that control applications share with most real-time computing applications. However, since control applications are reactive in nature it is more expressed for these. An overly pessimistic WCET estimation may cause the designer to chose a more powerful processor, which then will be under-utilized. Alternatively, the designer will reduce the task utilization by increasing the task periods, which will lead to poor control performance. In some control applications, e.g., hybrid and switching controllers and controllers employing on-line optimization, the computational workload can change dramatically over time as different control algorithms are switched in and out when the external environment changes, and from job to job due to the varying number of iterations that are needed in the optimization.

An optimization-based approach to feedback scheduling requires performance metrics that are parameterized with scheduling-related parameters, e.g., task periods. For general applications this is normally not available. However, for control application such performance metrics often exists. For example, using tools such as Jitterbug, [Lincoln and Cervin, 2002], it is possible to evaluate variance-type performance indices for linear control systems as a function of sampling periods.

5.1 Actuators & Sensors

The actuators of a feedback scheduler are the means with which the scheduler can modify the CPU utilization. For a controller task the task period is a natural actuator. Changing the task period dynamically may be more or less difficult depending on how the controller is implemented. For a controller implemented on input-output form with an internal state consisting of multiple lagged values of the measured value and the control signal it is generally more difficult to change the sampling period than for a controller that is realized on state-space form. In certain cases it may be necessary to use a Kalman filter to estimate the values of the state at the new sampling instants.

Depending on the performance requirements one can either adjust the controller parameters when the task period is changed, i.e., use gain scheduling with respect to the sampling interval, or use the same controller parameters independently on the task period. Another issue is the scheduling problem associated

with mode changes. Although the task set may be schedulable both before and after the mode change, it is not necessarily schedulable during the mode change.

An alternative actuator is the execution time demands of the controller. This can be realized using a multiple-versions approach or using an anytime approach. In a multiple version approach one may use multiple control algorithms with different execution time demands or one may occasionally skip the execution of parts of the control algorithm. The anytime approach can be applied to controllers in which the computations in the control algorithm or the sensors can be expressed in an iterative way that may be terminated after an arbitrary number of iterations, and where the control performance increases with the number of iterations. One example where this is applicable is model-predictive controllers (MPC) in which an quadratic optimization problem is solved in every sample. In the case of an overload, the optimization may be terminated early and still produce acceptable results. In [Henriksson *et al.*, 2004], value-based dynamic scheduling of multiple MPC controllers is considered using the optimization cost function as the value function.

Another example is vision sensors where the image processing can be organized to give improved object position estimates the more computation time that is available [Henriksson and Olsson, 2004]. However, in general, changing the task periods is more natural for control algorithms than changing the execution time demands.

The sensor in this type of feedback scheduler is a measurement of the actual CPU utilization. This assumes that the processor and RTOS are equipped with the means to perform such measurements. In order to avoid control actions caused by spurious measurement outliers ("noise") a low-pass filter may be included in the sensor. The low-pass filter can also be used to calculate an average of the utilization over a certain time period, e.g., the sampling period of the feedback scheduler.

5.2 Optimal Stationary Feedback Scheduling

The dynamics involved in feedback scheduling are often of low order or even purely static. The reason for this is obvious. If a task is given more or less CPU time the total utilization will change as soon as the next job of the task is started. Often the dynamics in the feedback loop comes from the filtering in the sensor. A consequence of this is that it is often enough with very simple control strategies in the feedback scheduler.

In [Cervin, 2003] it was shown that a simple linear proportional rescaling of the nominal task periods in order to meet the utilization set-point is optimal with respect to the stationary control performance under certain assumptions. It holds if the control cost functions, $J_i(h_i)$, where h_i is the sampling period, are quadratic, i.e.,

$$J_i(h_i) = \alpha_i + \beta_i h_i^2$$

or if they are linear,

$$J_i(h_i) = \alpha_i + \gamma_i h_i,$$

and if the objective of the feedback scheduler is to minimize the sum of the control cost functions or a weighted sum of the control cost functions. Linear or quadratic cost functions are quite good approximations of true cost functions in many cases.

The advantage of this approach is a simple and fast calculation that easily can be performed on-line. The linear rescaling also has the advantage that it preserves rate-monotonic ordering of the tasks and, thus, avoids any changes in task priorities in the case that rate-monotonic fixed priority scheduling is used. It is also possible to add more constraints to the optimization problem and still retain a simple solution. For example, one can use the nominal task periods as minimum task periods and use these whenever the utilization is less than the utilization set-point. However, the linear rescaling property does not hold in all cases. If the task set includes both tasks with quadratic cost functions and tasks with linear cost functions, the solution is not as simple, although it is still computable.

It is also possible to assign maximum sampling periods to certain tasks. This leads however to an iterative computation (LP-problem) in order to find the total rescaling of all the tasks. This is equivalent to the calculations needed in the elastic task model, [Buttazzo *et al.*, 1998], when the tasks (springs) have constraints on how much they may be compressed. However, it should be noted that the the cost functions above only concern the task periods and not the input-output latencies.

5.3 Optimal Feedback Scheduling

A drawback with the previous approach is that it does not consider the actual control performance. The optimization only concerns the stationary performance. Disturbances acting on the control loops will not be taken into account in the optimization. In [Henriksson and Cervin, 2005] an alternative approach is proposed. Instead of basing the optimization on stationary cost functions it is based on finite-horizon cost functions related to the sampling period, the current state of the control loop, and the period at which the feedback scheduler is invoked. The optimization horizon corresponds to the period of the feedback scheduler. Hence, rather than having cost functions that only are a function of the task periods, i.e.,

$$J_i(h_i) = \alpha_i + \beta_i h_i^k,$$

the cost functions now can be expressed as

$$J_i(h_i, x_i, T_{fbs}) = \alpha_i(x_i, T_{fbs}) + \beta_i(x_i, T_{fbs}) h_i^k.$$

The intuition behind this formulation is that a process in a transient phase, e.g., during a setpoint change, or exposed to an external disturbance may require more resources, e.g., a smaller sampling interval, than a process in stationarity.

The approach is formulated for linear quadratic (LQ) controllers, although the same approach also works for an arbitrary (i.e., non-optimal) state-feedback control law. The optimization objective is to minimize the combined performance of all the control loops,

$$\min_{h_1 \ldots h_n} \sum_{i=1}^{n} J_i(h_i, x_i, T_{fbs})$$

subject to the utilization bound given by the schedulability condition

$$\sum_{i=1}^{n} \frac{C_i}{h_i} \leq U_{sp}$$

This problem is a convex problem if the functions $J_i(1/f_i, x_i, T_{fbs})$ are convex in f_i. If all the cost functions have the same shape then an explicit solution

exists. If that is not the case analytical solutions exists only for special cases, e.g., integrator processes with minimum variance design cost functions. In other cases the cost functions are approximated as linear functions at the current sampling period and the cost function derivatives for each controller are computed off-line and stored in look-up tables.

An important design parameter is the feedback scheduler period. The shorter the period, the more responsive the system will be to external disturbances. However, the execution of the feedback scheduler induces overhead and consumes CPU time from the control tasks.

5.4 Feedback Scheduling Structures

Different structures are possible in feedback scheduling. A pure feedback scheme is reactive in the sense that the feedback scheduler will only remove a utilization error once it is already present. By combining the feedback with feedforward a pro-active scheme is obtained. The feedforward path could be use to allow controller task to inform the scheduler that they are changing their desired amount of resources, e.g., changing their execution times or nominal sampling periods, and to give the scheduler the possibility to compensate for this before any overload has occurred. The feedforward path can be also be used for dynamic task admission. A block diagram of the feedback-feedforward structure is shown in Fig. 1.

5.5 Value-Based Feedback Scheduling

Most of the feedback scheduling approaches proposed for control applications are indirect. By adjusting the task parameters, e.g., period and execution time, one makes sure that the task set is schedulable and has certain timing properties (latencies and jitter).These timing properties will then indirectly determine the performance of the application. The problem with this is the relationship between the timing parameters and the cost/performance. Often the relationship only holds in stationarity and in a mean-value sense.

An interesting but still largely unexplored approach is to instead use value-based or direct feedback scheduling. Here, the idea is to base the decision of which task to execute on an instantaneous cost function. This cost function should grow the longer the control loop executes in open loop and decrease when a control action is issued. The instantaneous cost could then be used as a dynamic task priority similar to the deadline in EDF. The resulting system would be a special case of an aperiodic event-triggered sampled system.

5.6 Research Directions

The challenges and resource directions for feedback scheduling of control tasks include all the challenges and research direction of control of CPU resources. Additionally, the following items are important:

- **Temporal robustness indices:** Indices are needed that allow us to decide how the control performance depends on the computing resources, e.g., on the sampling period. Although work has and is being done in this area more work is necessary
- **Resource negotiation frameworks:** Frameworks must be developed that allow dynamic negotiation about resources and control performance between the control applications and the QoS manager. These frameworks

must be able to express the control-specific aspects of the problem in addition to the computing and scheduling-specific aspects. It must also be able to express the performance requirements of the different control loops in a flexible way.

- **Formal performance guarantees:** Formal performance guarantees on control loops is something that today require that we have a fairly static implementation of the control system with respect to resource utilization, e.g., a statically scheduled time-triggered control loop has a very predictable performance. It is an open question whether it is possible to combine the flexibility implied by feedback scheduling with formal guarantees and, in that case, what type of formal guarantees.

6. Control Middleware

Applying control techniques to a computer software system requires certain generic services that are common to most applications. Often these have to do with the sensor and actuator interface of the control loop. For example, in queue delay control it is necessary to be able to measure arrival and departure times of requests and calculate average delays. Rather than implementing the support for this in every application (waste of development resources) or provide the support in the operating system (often not possible) an alternative is to use middleware technology.

A middleware is a software abstraction layer that mediates the interactions between a component or application and its environment or between two applications by providing services that the applications may call, i.e., the middleware provides the glue between the application and the interface. Middleware technology is commonly used in distributed system to provide communication services. Some examples are Java-RMI, Microsoft's COM, and CORBA. Middleware frameworks are also available for real-time and embedded systems, e.g., RT-CORBA and Embedded CORBA. There are also a large amount of research middleware framework developed for pervasive networked embedded system applications, e.g., mobile systems and sensor systems. Some examples of these are GAIA [Romn *et al.*, 2002], WSAMI [Issarny *et al.*, 2005], and AURA [Garlan *et al.*, 2002].

A few middleware solution have been developed explicitly for control purposes. ControlWare [Zhang *et al.*, 2002] is a middleware QoS-control architecture originally designed to help programmers apply control theory to control software performance. It allows the user to express QoS specifications off-line, maps these specifications into appropriate feedback loops. tunes the controllers to guarantee various performance specifications, and connects the loops to the right performance sensors and actuators in the application code such that the desired QoS is achieved [Abdelzaher *et al.*, 2003]. The aim of ControlWare is to isolate the software application programmer from control theoretic issues while still utilizing the theory. At the same time, it isolates the control engineer from the software task of interfacing the controller to the controlled software system and designing software performance sensors and actuators.

The basic abstraction provided in ControlWare is a component. A component has several input ports, output ports and some parameters. Components are connected via their ports, and communicate with each other via an infrastructure

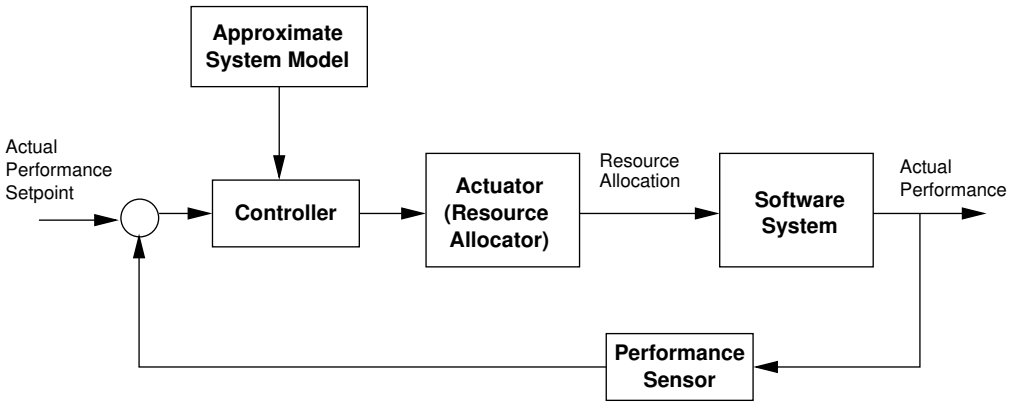

Figure 11 Absolute Guarantee Control Pattern (from [Zhang *et al.*, 2002])

named Softbus. Properly connected, several components (various number of sensors, actuators and controllers) form a control loop. Two main types of software sensors and actuators are supported: passive and active. A passive sensor or actuator is simply a function or software component that returns sensor data or accepts a command to perform an actuation when called by a controller. An active sensor, on the other hand, is a thread that usually is awakened periodically by the operating system to perform sensing or actuation.

The topology of a control loop is described by a template. Essentially, a template describes a general solution to a type of QoS guarantee. In other words, it maps a type of QoS guarantee problem into a single control problem. Several QoS performance control templates are supported. For example, absolute convergence guarantees, relative differentiated service guarantees, prioritization, and optimization guarantees.

The absolute convergence guarantee ensures that some performance metric R converges asymptotically to a desired value $R_{desired}$ and that the error is bounded at all times. The guarantee is translated into the control loop shown in Fig. 11. In [Zhang *et al.*, 2002] the corresponding control loop patterns for relative differentiated service, prioritization, and utility maximization are presented.

A related middleware example is IBM's Autotune Agents, see [Diao *et al.*, 2003] where an agent-based solution is proposed which automates the tuning of the IT environment for e-commerce applications and also automatically designs an appropriate tuning mechanism for the target system. The paper illustrates this in the context of managing a web server, where the problem of controlling CPU and memory utilization of an Apache web server is studied using the application-level tuning parameters MaxClients and KeepAlive which are exposed by the server. Using the AutoTune agent framework agents are constructed to automate a control-theoretic methodology that involves model building, controller design, and run-time feedback control. The designed feedback control system is able to handle the dynamic and interrelated dependencies between the tuning parameters and the performance metrics with guaranteed stability.

The Agilos (Agile QoS) architecture, [Li and Nahrstedt, 1999], is a middleware control architecture designed to provide middleware services to assist application-aware adaptations, namely, adaptation mechanisms that are tuned to the performance goals and specific functionalities of an application and which attempt to

adapt themselves or the applications for the purpose of providing the best possible QoS under available resource conditions, and of achieving the most graceful quality degradation in case of scarce resources. Agilos is designed as a three-tier architecture: In the first and lowest tier, application-neutral adaptors and observers maintain tight relationships with individual types of resources, and react to changes in resource availability. In the second tier, application-specific configurators are responsible for making decisions on when and what adaptive mechanisms are to be invoked in a client-server application, based on on-the-fly user preferences and application-specific rules. Furthermore, though each configurator corresponds to one application, configurators share the same fuzzy inference engine for rule processing. Finally, QualProbes provide QoS probing and profiling services so that application-specific adaptation rules can be either derived by measurements or specified explicitly by the user. In the third tier, a gateway and negotiators are introduced to control adaptation behavior in an application with multiple clients and servers, so that dynamic reconfigurations of client-server mappings are possible and tuned to the best interests of the application. The adaptation algorithm in Agilos is based on PID control.

FCS/nORB is a feedback control real-time scheduling service on nORB, a small-footprint Object Request Broker (ORB) middleware for networked embedded systems [Lu *et al.*, 2003]. FCS/nORB provides middleware support for real-time performance portability across platforms and robust performance guarantees in face of workload/platform variations. Three types of control loops are supported: control of CPU utilization, control of deadline miss ratio, and combined control of utilization and miss ratio.

The same group is also developing CAMRIT, a control-based adaptive middleware framework for real-time image transmission in distributed real-time embedded systems [Wang *et al.*, 2004]. CAMRIT features a distributed feedback control loop that meets image transmission deadlines by dynamically adjusting the quality of image tiles. Control theory is applied to design a control algorithm with analytic assurance of system stability and performance, despite uncertainties in network bandwidth.

There are also other types of middleware associated with control. However, the majority of these are intended for real-time control, i.e., control of some physical system using some type of networked embedded control system. One system worth mentioning, however, is Etherware [Baliga *et al.*, 2004]. Etherware is a messaging middleware for networked control loops. Of key importance is the concept of service continuity, i.e., the ability to maintain a communication channel during node restarts and upgrades and to recover from failure situations.

6.1 Research Directions

The most important research item for control middleware is to develop these systems from research prototypes to something that may be used more widely. Other research directions of importance are the following:

- **Middleware functionality:** It is still an open question whether the middleware only should be passive, i.e., provide sensing and actuation services that the application can use to itself implement the feedback control, or if it should be active, i.e., the middleware should be responsible for the actual control loop. Both of these approaches have advantages and disadvantages.

7. Control of Communication Networks

Traffic control of communication networks involves issues such as congestion control, routing and admission control. Here we survey the main areas relevant for this roadmap. Several other surveys are also available, e.g., see [Liu *et al.*, 2003a; Kwon and Kim, 2000]. Of particular interest is congestion control and how to control heterogeneous networks consisting of a blend of wired and wireless links.

7.1 Congestion Control

The success of the Internet as a worldwide information carrying network can be attributed to the feedback mechanisms that controls the data transfer in the transport layer in the Internet protocol stack. These algorithms has historically managed to distribute network resources among contending users in a sufficiently fair and resource-efficient way. An explanation to this is that the control is allocated at the end-systems (users) and hence obey a decentralized structure. This design allows widely heterogeneous demands ranging from a few packets ("mice") to long bandwidth greedy streams ("elephants"), but still avoids the complexity of a centralized allocation mechanism. The Transmission Control Protocol (TCP) that was presented in the late 1980s [Jacobson, 1988] and its numerous refinements, see e.g. [Stevens, 1997; Mathis *et al.*, 1996; Hoe, 1996; Floyd and Henderson, 1999; Jacobson *et al.*, 1992; Allman *et al.*, 1999], is the dominating end user protocol used today carrying approximately 90% of the total traffic.

Furthermore, together with the source control, buffers have played a key-role during the evolution of the Internet. Since end-users base control action on limited, corrupt and delayed information; buffers are used at links inside the network to smooth out errors in the control, hence making the system more robust. Auxiliary control from the network interior has also been introduced by "intelligent" links that marks or drop packets depending on the traffic load. This is referred to as Active Queue Management (AQM) in the literature, see e.g. [Hollot *et al.*, 2001a] and the references therein for examples.

Historically congestion control algorithms have been designed by computer scientist outside the framework of control theory. The tremendous complexity of the Internet makes it extremely difficult to model and analyze, and it has been questioned if mathematical theory can offer any major improvements in this area. However, recently significant progress in the theoretical understanding of network congestion control has been made following seminal work by Kelly and coworkers [Kelly *et al.*, 1998; Kelly, 1999] (see also the surveys [Kelly, 2003; Low and Srikant, 2004] and the book [Srikant, 2004]). The key is to work at the correct level of aggregation which is fluid flow models with validity at longer time-scales than the round-trip time (RTT). By explicitly model the congestion measure signal fed back to sources, posing the network flow control as an optimization problem where the objective is to maximize the total source utility, it is shown that the rate control problem can be solved in a completely decentralized manner [Kelly *et al.*, 1998; Low and Lapsley, 1999] under the constraint that each source has a (concave) utility function of its rate. The aggregated congestion measure along the path, which can be packet loss probability or queuing delay depending on the protocol variant, corresponds to the Lagrange multipliers in the optimization (or price in an economic interpretation) and has to be distributed to, or estimated, at the end-users. The pricing algorithm is carried out by AQM at the individual links.

This optimization perspective of the rate control problem has been taken in a number of contributions. It also allow for dynamical laws and the developed algorithms can be classified as (1) primal, when the control at the source is dynamic but the link uses a static law; (2) dual, when the link uses a dynamic law but the source control is static; and (3) primal–dual, when dynamic controls are used both at the source and the links, see [Low *et al.*, 2002a; Liu *et al.*, 2003b; Low and Srikant, 2003], for nice overviews.

By appropriate choice of utility function, even protocols not based on optimization, such as TCP Reno, can be interpreted as distributed algorithms trying to maximize the total utility [Low and Srikant, 2003; Low, 2000]. Delay based protocols such as TCP Vegas [Brakmo and Peterson, 1995] or TCP FAST [Jin *et al.*, 2004] can be classified as a primal–dual algorithm with the queuing delay as a dynamic link price which is estimated at the source.

To ensure that the system will reach and maintain a favorable equilibrium, it is important to assess the dynamical properties, such as stability and convergence, of the schemes. Instability means that the protocol is unable to sustain the equilibrium, and manifests itself as severe oscillations in aggregate traffic quantities, such as queue lengths.

Stability of the basic schemes, which allow dynamic rate control and static marking, or dynamic queue management schemes and static source rate control, was established already in [Kelly *et al.*, 1998; Low and Lapsley, 1999] but under very idealized settings. When both source rate and link price updates are dynamic, stability has been proved using time-scale separation in [Kunniyur and Srikant, 2002], and for the single bottleneck case in [Altman *et al.*, 1998; Hollot and Chait, 2001]. A unifying framework for establishing global stability of congestion control laws based on passivity has been proposed in [Wen and Arcak, 2004].

The above results have all ignored the effect of network delay, and assumed that price information is available instantaneously at the source, that the sources take immediate action, and that the new rates affect the link prices instantaneously. However, stability of the protocols in equilibrium depends critically on the feedback delay. Naturally, recent research focus on source- and link control laws that guarantees stability for more general network configurations and delay distributions. Conditions for local stability of a single-user, single bottleneck scenario were derived in [Johari and Tan, 2001], and it was conjectured that the same condition guarantees stability also in the case of heterogeneous round-trip delays. A weaker version of the conjecture was proved in [Massoulie, 2000] and the original conjecture was proved in [Vinnivombe, 2000; Vinnicombe, 2002]. Local stability of Reno/RED with feedback delays has been studied in [Hollot *et al.*, 2001; Low *et al.*, 2002b]. The stability analysis reveals that these protocols tend to become unstable when the delay increases and, more surprisingly, when the capacity increases. This has spurred an intensive research in protocols that maintains local stability also for networks with high bandwidth-delay product, see e.g., [Paganini *et al.*, 2001; Floyd, 2003; Kelly, 2003]. In [Paganini *et al.*, 2001] the authors study decentralized control that scales with network capacity and proves local stability for heterogeneous delays. This class of controllers are further examined in [Lestas and Vinniecombe, 2004] and it is stated that the stability result is valid even for the less ideal case of non-symmetric protocols.

Local stability results have mainly been achieved using frequency methods, a major strength is that they take delay into account. However, classical Lyapunov

techniques, used for analyzing global stability properties, do not have this feature. To be able to achieve global results but still not ignoring delay one have to rely on alternative methods such as e.g. the Lyapunov-Krasovskii and the Lyapunov-Razumikhin methods [Gu *et al.*, 2003; Niculescu, 2001], which are closely related extensions of Lyapunov's classical method. A further discussion on different methods for network analysis can be found in [Papachristodoulou *et al.*, 2004]. Moreover, in [Deb and Srikant, 2003] the Lyapunov-Razumikhin theorem is used to achieve conditions for global stability for a single link accessed by a single source in the presence of delay. The same technique is used in [Ying *et al.*, 2004] to generalize the result to a general topology and multiple sources with heterogeneous delays. Lyapunov-Krasovskii functionals is used in [Mazenc and Niculescu, 2003] to study global stability of a class of nonlinear dynamical systems with delay presented by Kelly [Kelly, 2001] and used to model congestion control mechanisms for the Internet. An alternative approach is taken in [Peet and Lall, 2004] where the authors analyzes the global stability of TCP/AQM setting over a single link with homogeneous delays. The stability proof is here based on the theory of integral-quadratic constraints.

7.2 Control and optimization of wireless networks

Future wireless networks are expected to support a wide variety of applications, ranging from high data rate services for flexible ad-hoc networks to ultra-low power operation for longevity of wireless sensor networks. Whereas the link capacities in wireline networks are fixed, the capacities of wireless links can be adjusted by the allocation of communications resources, such as transmit powers, bandwidths, or time-slot fractions, to different links. Adjusting the resource allocation changes the link capacities, influences the optimal routing of data flows, and alters the total utility of the network. Hence, the optimal network operation can only be achieved by coordinating the operation across the networking stack. This is often referred to as *cross-layer optimization*. Emerging microprocessor technologies are enabling wireless units to become equipped with the processing power needed to implement adaptive transmission techniques and to make intelligent decisions about packet routing and resource management – cross-layer coordination is becoming technologically feasible.

A fundamental question is whether it is worthwhile to introduce advanced resource management and coordination schemes. One way of attacking this problem is to try to determine the *information-theoretic capacity*, which includes optimization over all possible modulation and coding schemes and involves many of the unsolved problems of network information theory [Ephremides and Hajek, 1998; Cover and Thomas, 1991]. Recent contributions in this direction can be found in, *e.g.*, [Gupta and Kumar, 2000; Grossglauser and Tse, 2001]. An alternative approach is to focus on *network layer capacity*, where coding and modulation schemes are fixed, and one optimizes over some critical parameters, such as power allocations and scheduling decisions. An initial study of the potential performance benefits of cross-layer coordination in a number of small ad-hoc networks was carried out in [Toumpis and Goldsmith, 2002], and models and methods for cross-layer optimization of multi-hop wireless networks have been proposed in, *e.g.*, [Xiao *et al.*, 2002; Julian *et al.*, 2002; Johansson *et al.*, 2003; Värbrand *et al.*, 2003; Cruz and Santhanam, 2003; Johansson and Xiao, 2004; Radunovic and Boudec, 2004]. These methods allow us to evaluate the cross-layer optimized performance of networks of significant sizes under orthogonal channel models [Xiao *et al.*, 2002], CDMA with and without [Julian *et al.*,

2002; Johansson *et al.*, 2003; Loretti *et al.*, 2005] interference cancellation, S-TDMA [Värbrand *et al.*, 2003; Johansson and Xiao, 2004] and ultra-wide band channels [Radunovic and Boudec, 2004]. While these references focus on performance and fairness objectives, power-optimal network operation is considered in [Cruz and Santhanam, 2003; Madan *et al.*, 2005].

The centralized optimization schemes described above are useful for gaining insight in the performance benefits of coordinating the different layers of the protocol stack, but are quite far from the distributed routing and resource management protocols needed in practice. Centralized solutions tend to incur large communication overhead costs, introduce a single-point-of failure, and scale poorly with the number of network nodes. Moreover, the optimal policies may be computationally demanding to execute. One way to synthesize a distributed protocol from a network model is to use mathematical decomposition techniques, similarly to the congestion control analysis and design methods for the fixed Internet [Low and Lapsley, 1999]. For example, by applying dual decomposition to the central optimization problem for wireless systems, one will often find that it is possible to subdivide the problem into smaller problems; typically a network problem (the same as the source and router subproblems for wireline networks) and an additional resource allocation subproblem. Whether or not the resource allocation subproblem can be solved in a distributed way depends on the channel models and the structure of the resource constraints; if the resource constraints are local to nodes and the channels are orthogonal, the problem is easily solved. With global constraints on the resources or with significant interference between channels, the problem gets harder, and a combination of these constraints are not likely to have a distributed solution. However, with only one of these constraints the problem can be solved in a distributed way. A distributed solution to the joint congestion control and power allocation problem for CDMA systems, under the assumption that all links can sustain high SINRs and that power constraints are local to each node, is presented in [Chiang, 2005]. A distributed solution to the joint congestion control and spectrum assignment problem for orthogonal channels but a network-wide resource constraint is derived in [Johansson and Johansson, 2005]. A particular feature of this algorithm is that nodes only negotiate and exchange resources with its neighbors. Distributed solutions to joint congestion control and link scheduling under simplified interference models are presented in [Chen *et al.*, 2005; Yi and Shakkottai, 2004].

All approaches described so far, centralized as well as distributed, have considered fluid models where traffic and link capacities are averaged. Cross-layer optimization under statistical traffic models have been considered in, *e.g.*, [Neely *et al.*, 2003; Lin and Shroff, 2004]

7.3 Interactions between TCP and wireless links

TCP was designed with wired links in mind, and performance problems are common when running TCP over wireless links. To address the problems, there are three main approaches.

Making the link friendlier to TCP In wired networks, almost all losses are due to congestion. The most fundamental problem with wireless links is that there are also a fairly high rate of losses due to noise and interference on the radio channel. One approach is to repair these losses at the link layer, using Forward Error Correction (FEC) or Automatic Repeat Request (ARQ), which retransmits damaged radio frames.

Forward error correction adds redundant data to the radio frames or sequences of radio frames, and the radio link receiver can use this redundancy to reconstruct the data even if part of the transmitted data is damaged. The cost is that some of the bandwidth is "wasted" on redundant data. The optimal amount of redundancy is a non-trivial trade-off [Chahed *et al.*, 2003; Barakat and Altman, 2002].

Automatic Repeat Request is a common mechanism in existing systems such as UMTS, and comes in several different flavours. This is a link-layer feedback mechanism where the radio link receiver detects damaged frames and asks the sender to retransmit them [Bai *et al.*, 2000; Canton and Chahed, 2001; Bertsekas and Gallager, 1992; Chockalingam *et al.*, 1999].

It is important to not focus only on the loss rate of the radio link, also the delay distribution can have an impact on TCP. The ARQ mechanism reduces the loss rate to almost zero, but instead adds random delays [Möller and Johansson, 2003; Möller *et al.*, 2004; Klein *et al.*, 2004].

Improvements to the TCP algorithm A different approach is to address the TCP algorithms, and try to make them more robust with respect to the transmission properties (primarily loss and delay distribution) of an heterogeneous network with mixed wired and wireless links. This is a huge area of current research [Mascolo *et al.*, 2001; Sarolahti *et al.*, 2003; Cen *et al.*, 2003; Samaraweera, 1999; Fu and Liew, 2003; Ludwig and Katz, 2000].

The challenge for end-to-end transport control is to estimate the important features of the network path, such as the roundtrip time, available bandwidth, bottleneck queue size, and to distinguish between events such as congestion loss, radio link losses, and temporary outages when packets are not lost but buffered somewhere in the network.

The goal of this line of research is not only to make TCP work well over a particular radio link technology, but to make TCP robust enough to work over a wide range of current and future radio links.

Split-connection The third approach puts a proxy in the network, close to the base station. The straight forward way to arrange this is to split the connection in two; one TCP connection between server and proxy (assumed to use the wired Internet), and another TCP connection between the proxy and the terminal [Bakre and Badrinath, 1995; Yavatkar and Bhagawat, 1994].

A split-connection setup violates the traditional end-to-end principle of the Internet. It can be attractive from a deployment perspective, because one can use a specialized transport protocol between the proxy and the terminal, tailored to the radio network at hand, without any changes to the TCP implementation in the server.

A more subtle form of proxying is to use a Protocol Helper. Such a proxy does not play the part of a TCP end point. Instead it monitors the packets that are part of each TCP stream, and it can add, manipulate, resort, duplicate, drop or delay packets, both data packets and acknowledgements.

The protocol helper hides radio related errors by locally resending lost downstream packets, or manipulating the upstream ack packets to avoid that the sending TCP interpret radio link errors as congestion losses.

7.4 Challenges and Research Directions

Control-based approaches in communication networks is a very large research field, in particular if wireless systems, e.g., sensor networks, are included. A necessarily non-exhaustive list of important research directions includes the following items:

- **Cross-layer design:** In order to be able to control the network performance it is necessary to measure and modify the network parameters. The current ISO-OSI stack layer is not ideally supported for cross layer designs where information from the lower layers must be made available at the application layer and where the application layer must be able to modify the behaviour of the lower layer protocols dynamically. Hence, new protocols and protocol models are needed that simplify this.

- **Theory and engineering principles for designing dynamic resource allocation protocols** Theories and engineering principles for dynamically allocating resources in wireless ad hoc networks to ensure quality of service are needed for a wide range of applications. One interesting suggestion is to have a formal, possibly optimization-based, theory for the design of network protocols based on a model of the underlying network and a specification of the application requirements. The theory should consider the effects and interactions of link layer, network layer, transport layer and the application, and ensure robust behavior in the presence of system variations and information delays.

- **Network State Estimation:** The control of network performance often requires access to network state variables, such as available bandwidth, round-trip times, and packet loss. These variables are typically not immediately available, but must be estimated from other quantities. The design of reliable and efficient estimators for network state is thus instrumental for many applications, and requires the development of simple and flexible models of network dynamics together with the associated advances in estimation theory.

- **Network interfaces:** Cross-layer feedback and designs pose requirements on the interface between the application layer and the physical layers. Which are the network sensors that can be used by an application to obtain information about the status of the network and its resources, i.e., link quality, power levels, local link utilization/delay, and retransmission/loss rates. Which are the corresponding network actuators that the application may use to effect the network performance? The actuators can, e.g., be divided into load management actuators (routing, load balancing, radio range control, packets drops etc), resource management actuators (node mobility, processor duty cycle etc), and, in particular in the case of sensor networks, data manipulation actuators (level of data aggregation, amount of in network processing, etc).

- **Interoperability of heterogeneous networks:** In the future our networks will be more and more heterogeneous, consisting of a mixture of different wired and wireless communication protocols. Is it possible to derive a theory for, e.g., network performance control, that is applicable also in this case?

8. Error Control of Software

The complexity and size of the software systems that our society depends on are continuously growing. The use of systems of systems and components developed independently and analyzed and tested in isolation, easily give rise to brittle and error-prone systems. Some of the sources of difficulties are unexpected interactions, inadequate development infrastructures, and system instabilities. Unexpected interactions are caused by incompatible abstractions, incorrect or implicit assumptions in the system interfaces, and incompatible real-time, fault-tolerance, and security protocols. Inadequate development infrastructures are reflected in the lack of domain-specific reference architectures, tools, and design patterns, with known and parameterized real-time, robustness, and security properties. Finally, system instabilities result when faults and failures in one component cascade along complex and unexpected dependency graphs resulting in catastrophic system-wide failures.

The solution to the problem of unexpected interactions is to provide techniques and tools that support making the semantic assumptions of each component explicit and machine checkable. The development infrastructure problem needs to be approached by the development of formally specified and validated coherent real-time, robustness, security, and networking protocols together with domain models, reference architectures, and tools with parameterized real time, robustness, and security properties. The solution to system instability is to focus on the development on stable software architectures. This is the core idea behind error control of software.

The development of completely defect-free complex software systems is extremely difficult, if not impossible. At the same time several large existing software systems are remarkably stable and reliable in the presence of thousands or maybe millions of residual software bugs, e.g., the telecom networks or the WWW system of systems. Hence, rather than focusing the development effort on trying to eliminate all bugs at design time it is important to develop methods that allow us to develop safe and stable software systems that still can utilize COTS-quality software components with a considerable amount of residual bugs. Hence, the focus should be on detection and recovery from software errors at run-time, in addition to elimination of software errors at design-time. With this approach the chances of developing robust software systems within finite budgets will be greatly increased, also for safety-critical applications.

The idea behind error control of software is to use ideas similar to the ideas used in feedback control in order to detect malfunctioning software components and, in that case fall back on, a well-tested core software component that is able to provide the basic application service with guarantees on performance and safety. Hence, the basic idea assumes that a certain amount of defect-free components are available, that can be used to implement the fall-back safety core service. The second key idea is to always design your system to have a simple and well formed dependency tree, with a minimal number of dependency relations among components and a maximal number of USE relations. This is necessary in order to be able to identify the core services and keep them small. The background to several of the key ideas of the area is given in [Sha, 2001].

One of the first examples of software error control is the Simplex architecture developed by Lui Sha et al at SEI/CMU, see, e.g., [Sha, 1998; Seto *et al.*, 1998]. The

objective of Simplex was to support safe on-line updates of safety-critical real-time control systems. The basic building blocks of Simplex were replacement units, software components with a given communication interface. The replacement units were organized into application units, which also included communication and management functions. An executing control system would typically consist of one safety unit (controller) and one baseline unit (controller). The safety unit guarantees a basic service quality level and contains operation monitoring code for monitoring the state of the application. The baseline controller is the controller that is executing normally. When the baseline controller should be updated with a new version, the execution of the new, possibly error-prone, baseline version is monitored by the safety unit. The monitoring includes both safety, i.e., is the system under control in a safe state, and performance, i.e., does the system perform no worse than with the old version of the baseline unit. If this is not the case the safety unit is switched in and, eventually, the old version of the baseline unit. The safety unit scheme provides protection against semantic application faults. This is combined with protection against timing faults through the use of watchdog timers, and protection against execution-related faults through enforcing that each replacement unit uses separated and protected data, storage, and computation resources.

A stable software system is a system that has bounded response to errors and which can maintain key properties in spite of errors in non-critical components. The domain of convergence in software error control is the states that satisfy the precondition of the recovery units. Stability control is the mechanism used to ensure that the preconditions will hold. A stable system allows for safe testing of new components under realistic operating conditions, i.e., on-line upgrades.

The reason why error control of software is treated in the context of this roadmap is our strong belief that real-time computing has a lot to learn and gain from control theory. However, in software error control our view of what control is has to be broadened substantially. Control is normally concerned with the temporal behaviour of systems. The ideas behind software error control are, however, not restricted to the temporal behaviour. The same approach can in principle also be used for applications that only contain functional requirements. In this case software error control has strong relationships to techniques that are commonly associated with fault tolerance, e.g., hardware and software redundancy and diversity through replication and N-version programming. However, the principles behind software error control have so far mainly been applied to reactive applications, i.e., avionics control systems. The major challenges is to develop a new paradigm for software stability control based on an integration of concepts from fault-tolerant computing and control that is applicable to a wide range of application types.

8.1 Research Directions

- **Application identification:** So far the software error control ideas have primarily been applied to reactive feedback control applications. It is necessary to investigate what other types of applications the approach is suitable for.
- **Software robustness/stability as a control problem:** The notion of feedback control needs to be broadened in order to really match the requirements of software error control. New formalisms and models for software error control are needed.

- **Application development:** The number of documented examples where software error control has been applied is very small. In order to increase the understanding for the subject and to develop the necessary methods and theory, more documented applications must be developed.
- **Relationships to ordinary fault-tolerance:**The relationships to the methods within the traditional fault tolerance area must be clarified.

9. References

Abdelzaher, T., J. Stankovic, C. Lu, R. Zhang, and Y. Lu (2003): "Feedback performance control in sofware services." *IEEE Control Systems Magazine*, **23:3**.

Abeni, L. and G. Buttazzo (1998): "Integrating multimedia applications in hard real-time systems." In *Proc. 19th IEEE Real-Time Systems Symposium*. Madrid, Spain.

Abeni, L. and G. Buttazzo (2001): "Hierarchical QoS management for time sensitive applications." In *Proc. IEEE Real-Time Technology and Applications Symposium*. Taipei, Taiwan.

Abeni, L. and G. Buttazzo (2004): "Resource reservation in dynamic real-time systems." *Real-Time Systems*, **27**, pp. 123–167.

Abeni, L., L. Palopoli, G. Lipari, and J. Walpole (2002): "Analysis of a reservation-based feedback scheduler." In *Proc. 23rd IEEE Real-Time Systems Symposium*.

Agnew, C. E. (1976): "Dynamic modeling and control of congestion-prone systems." *Operations Research*, **24:3**, pp. 400–419.

Allman, M., v. Paxson, and W. Stevens (1999): "TCP congestion control." RFC 2581.

Altman, E., T. Basar, and R. Srikant (1998): "Robust rate control for ABR sources." In *IEEE Infocom*, pp. 166–173. San Francisco, CA.

Amirijoo, M., J. Hansson, S. Gunnarsson, and S. Son (2005): "Enhancing feedback control scheduling performance by the on-line quantification and suppression of measurement disturbance." In *Proceedings of the 11th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'05)*.

Amirijoo, M., J. Hansson, and S. Son (2003a): "Algorithms for managing QoS for real-time data services using imprecise computation." In *Proceedings of the 9th International Conference on Real-Time and Embedded Computing Systems and Applications (RTCSA)*.

Amirijoo, M., J. Hansson, and S. Son (2003b): "Error-driven QoS management in imprecise real-time databases." In *Proceedings of the 15th EuroMicro Conference on Real-Time Systems (ECRTS)*.

Amirijoo, M., J. Hansson, and S. Son (2003c): "Specification and management of QoS in imprecise real-time databases." In *Proceedings of the Seventh International Database Engineering and Applications Symposium (IDEAS'03)*.

Amirijoo, M., J. Hansson, and S. H. Son (2003d): "Specification and management of qos in imprecise real-time databases." In *Proceedings International Database Engineering and Applications Symposium*, pp. 192–201.

Aparah, D. (1998): "Adaptive resource management in a multimedia operating system." In *Proceedings of the 8th International Workshop on Network and Operating System Support for Digital Audio and Video*.

Åström, K. J. and B. Bernhardsson (1999): "Comparison of periodic and event based sampling for first-order stochastic systems." In *Preprints 14th World Congress of IFAC*, vol. J, pp. 301–306.

Åström, K. J. and T. Hägglund (1995): *PID Controllers: Theory, Design, and Tuning*. Instrument Society of America, Research Triangle Park, North Carolina.

Åström, K. J. and B. Wittenmark (1995): *Adaptive Control*. Addison-Wesley, Reading, Massachusetts.

Åström, K. J. and B. Wittenmark (1997): *Computer-Controlled Systems*. Prentice Hall.

Bai, Y., P. Zhu, A. Rudrapatna, and A. T. Ogielski (2000): "Performance of TCP/IP over IS-2000 based CDMA radio links." In *Proc. of IEEE 52th VTC'2000-Fall*. IEEE.

Bakre, A. and B. R. Badrinath (1995): "I-TCP: Indirect TCP for mobile hosts." *15th International Conference on Distributed Computing Systems*.

Baliga, G., S. Graham, L. Sha, and P. R. Kumar (2004): "Service continuity in networked control using Etherware." In *Proceedings of Middleware 2004, Toronto, Canada*.

Barakat, C. and E. Altman (2002): "Bandwidth tradeoff between TCP and link-level FEC." *Comput. Networks*, **39:5**, pp. 133–150.

Bertsekas, D. and R. Gallager (1992): *Data Networks*, second edition edition. Prentice Hall.

Blevins, P. and C. Ramamoorthy (1976): "Aspects of a dynamically adaptive operating system." *IEEE Transactions on Computers*, **25:7**, pp. 713–725.

Bouyssounouse, B. and J. Sifakis, Eds. (2005): *Embedded Systems Design: The ARTIST Roadmap for Reasearch and Development*. Number 3436 in LNCS. Springer-Verlag.

Brakmo, L. S. and L. L. Peterson (1995): "TCP Vegas: end-to-end congestion avoidance on a global Internet." *IEEE Journal on Selected Areas in Communications*, **13:8**, pp. 1465–1480.

Burns, A., D. Prasad, A. Bondavalli, F. D. Giandomenico, K. Ramamritham, J. Stankovic, and L. Stringini (2000): "The meaning and role of value in scheduling flexible real-time systems." *Journal of Systems Architecture*, **46**, pp. 305–325.

Buttazzo, G., G. Lipari, and L. Abeni (1998): "Elastic task model for adaptive rate control." In *Proc. 19th IEEE Real-Time Systems Symposium*, pp. 286–295.

Canton, A. and T. Chahed (2001): "End-to-end reliability in UMTS: TCP over ARQ." In *Globecom 2001*.

Cassandras, C. and S. Lafortune (1999): *Introduction to Discrete Event Systems*. Kluwer.

Cen, S., P. C. Cosman, and G. M. Voelker (2003): "End-to-end differentiation of congestion and wireless losses." *IEEE/ACM Trans. on Networking*, **11:5**, pp. 703–717.

Cervin, A., J. Eker, B. Bernhardsson, and K.-E. Årzén (2002): "Feedback-feedforward scheduling of control tasks." *Real-Time Systems*, **23:1**.

Cervin, A., D. Henriksson, B. Lincoln, J. Eker, and K.-E. Årzén (2003): "How does control timing affect performance?" *IEEE Control Systems Magazine*, **23:3**, pp. 16–30.

Chahed, T., A.-F. Canton, and S.-E. Elayoubi (2003): "End-to-end TCP performance in W-CDMA/UMTS." In *ICC'2003*. Anchorage.

Chen, L., S. H. Low, and J. C. Doyle (2005): "Joint congestion and media access control design for ad hoc wireless networks." In *Proceedings of the IEEE Infocom*. Miami, FL. To Appear.

Chiang, M. (2005): "Balancing transport and physical layers in wireless multihop networks: Jointly optimal congestion control and power control." *IEEE Journal on selected areas in communications*, **23:1**, pp. 104–116.

Chockalingam, A., A. Zorzi, and V. Tralli (1999): "Wireless TCP performance with link layer FEC/ARQ." In *Proceedings of IEEE ICC'99*, pp. 1212–1216.

Christin, N., J. Liebeherr, and T. Abdelzaher (2002): "A quantitative assured forwarding service." In *Proceedings of IEEE INFOCOM*.

Chu, H. and K. Nahrstedt (1999): "CPU service classes for multimedia applications." In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*.

Cover, T. M. and J. A. Thomas (1991): *Elements of Information Theory*. John Wiley & Sons, Inc., New York.

Crovella, M. E. and A. Bestavros (1997): "Self-similarity in world wide web traffic: Evidence and possible causes." *ACM/IEEE Transaction on Networking*, **5:6**.

Cruz, R. L. and A. V. Santhanam (2003): "Optimal routing, link scheduling and power control in multi-hop wireless networks." In *Proceedings of the IEEE Infocom*. San Francisco, CA.

Cucinotta, T., L. Palopoli, and L. Marzario (2004): "Stochastic feedback-based control of QoS in soft real-time systems." In *Proceedings of the Conference on Decision and Control (CDC)*.

Deb, S. and R. Srikant (2003): "Global stability of congestion controllers for the internet." *IEEE Transactions on Automatic Control*, **48:6**, pp. 1055–1060.

Diao, Y., N. Gandhi, J. Hellerstein, S. Parekh, and D. Tilbury (2002): "Mimo control of an apache web server: Modeling and controller design." In *American Control Conference*.

Diao, Y., J. L. Hellerstein, S. Parekh, and J. P. Bigus (2003): "Managing web server performance with autotune agents." *IBM Systems Journal*, **42:1**, pp. 136–149.

ElBatt, T. and A. Ephremides (2002): "Joint scheduling and power control for wireless ad-hoc networks." In *Proceedings of the IEEE Infocom*. New York, NY.

Ephremides, A. and B. Hajek (1998): "Information theory and communication networks: an unconsummated union." *IEEE Transactions on Information Theory*, **44:6**, pp. 2416–2434.

Floyd, S. (2003): "Highspeed TCP for large congestion windows." Internet Draft `<draft-floyd-tcp-highspeed-02.txt>`.

Floyd, S. and T. Henderson (1999): "The NewReno modification to TCP's fast recovery algorithm." RFC 2582.

Fu, C. P. and S. C. Liew (2003): "TCP veno: TCP enhancement for transmission over wireless access networks." *IEEE Journal on Selected Areas in Communications*, **21:2**, pp. 216–228.

Gandhi, N., S. Parekh, J. Hellerstein, and D. Tilbury (2001): "Feedback control of a lotus notes server: Modeling and control design." In *American Control Conference*.

Garlan, D., D. P. Siewiorek, A. Smailagic, and P. Steenkiste (2002): "Aura: Toward distraction-free pervasive computing." *IEEE Pervasive Computing*.

Gill, C. D., D. L. Levine, and D. C. Schmidt (1998): "Dynamic scheduling strategies for avionics mission computing." In *Proc. 17th IEEE/AIAA Digital Avionics Systems Conference*.

Grossglauser, M. and D. Tse (2001): "Mobility increases the capacity of ad-hoc wireless networks." In *Proceedings of the IEEE Infocom*. Anchorage, AL.

Gu, K., V. L. Kharitonov, and J. Chen (2003): *Stability of Time-Delay Systems*. Birkhäuser.

Gupta, P. and P. R. Kumar (2000): "The capacity of wireless networks." *IEEE Transactions on Information Theory*, **46**, March, pp. 388–404.

Hajek, B. and G. Sasaki (1988): "Link scheduling in polynomial time." *IEEE Transactions on Information Theory*, **34:5**, pp. 910–917.

Hellerstein, J. L., Y. Diao, S. Parekh, and D. M. Tilbury (2004): *Feedback Control of Computing Systems*. John Wiley.

Henriksson, D. and A. Cervin (2005): "Optimal on-line sampling period assignment for real-time control tasks based on plant state information." In *Proceedings of the Joint IEEE CDC–ECC Conference, Sevilla, Spain, December 2005.*

Henriksson, D., Y. Lu, and T. Abdelzaher (2004): "Improved prediction for web server delay control." In *submission to Euromicro Conference on Real-Time Systems*. Catania, Sicily, Italy.

Henriksson, D. and T. Olsson (2004): "Maximizing the use of computational resources in multi-camera feedback control." In *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS04)*. Toronto, Canada.

Hoe, J. C. (1996): "Improving the start-up behavior of a congestion control scheme for TCP." In *Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*, pp. 270–280.

Hollot, C., V. Misra, D. Towsley, and W.-B. Gong (2001): "A control-theoretic analysis of RED." In *IEEE Infocom*, vol. 3, pp. 1510–1519. Anchorage, Alaska.

Hollot, C. V. and Y. Chait (2001): "Nonlinear stability analysis for a class of TCP/AQM networks." In *IEEE CDC*, pp. 2309–2314. Orlando, FL.

Hollot, C. V., V. Misra, D. F. Towsley, and W. Gong (2001a): "A control theoretic analysis of RED." In *Proceedings of IEEE INFOCOM*, pp. 1510–1519.

Hollot, C. V., V. Misra, D. F. Towsley, and W. Gong (2001b): "On designing improved controllers for AQM routers supporting TCP flows." In *Proceedings of IEEE INFOCOM*, pp. 1726–1734.

Iijima, T., K. Ouchi, Y. Maruyama, and S. Nemoto (2002): "Hitachi's latest supervisory and control system for advanced combined cycle power plants." *Hitachi Review*.

Issarny, V., D. Sacchetti, F. Tartanoglu, Ferdaand Saihan, R. Chibout, N. Levy, and A. Talamona (2005): "Developing ambient intelligence systems: A solution based on web services." *Journal of Automated Software Engineering*, **12**.

Jacobson, V. (1988): "Congestion avoidance and control." *SIGCOMM Comput. Commun. Rev.*, **18:4**, pp. 314–329.

Jacobson, V., R. Braden, and D. Borman (1992): "TCP extensions tor high performance." RFC 1323.

Jin, C., D. X. Wei, and S. H. Low (2004): "FAST TCP: motivation, architecture, algorithms, performance." In *Proceedings of IEEE Infocom*. IEEE.

Johansson, B. and M. Johansson (2005): "Primal and dual approaches to distributed cross-layer optimization." In *16th IFAC World Congress on Automatic Control*. To Appear.

Johansson, M. and L. Xiao (2004): "Scheduling, routing and power allocation for fairness in wireless networks." In *Proceedings of IEEE VTC Spring*. Milan, Italy.

Johansson, M., L. Xiao, and S. Boyd (2003): "Simultaneous routing and power allocation in CDMA wireless networks." In *Proceedings of the IEEE International Conference on Communications*, pp. 51–55. Anchorage, Alaska.

Johari, R. and D. Tan (2001): "End-to-end congestion control for the Internet: delays and instability." *IEEE/ACM Transactions on Networking*, **6:9**, pp. 818–832.

Julian, D., M. Chiang, D. O'Neill, and S. Boyd (2002): "QoS and fairness constrained convex optimization of resource allocation for wireless cellular and ad hoc networks." In *Proceedings of the IEEE Infocom*, pp. 1–10. New York, NY.

Kelly (2001): "Mathematical modelling of the internet." In *Bjorn Engquist and Wilfried Schmid (Eds.), Mathematics Unlimited – 2001 and Beyond@ Springer*.

Kelly, F. (2003): "Fairness and stability of end-to-end congestion control." *European Journal of Control*, **9**, pp. 159–176.

Kelly, F., A. Maulloo, and D. Tan (1998): "Rate control in communication networks: shadow prices, proportional fairness and stability." *Journal of the Operational Research Society*, **49**, pp. 237–252.

Kelly, F. P. (1999): "Mathematical modelling of the Internet." In *Fourth International Congress on Industrial and Applied Mathematics, Edinburgh, Scotland*.

Kelly, T. (2003): "Scalable TCP: improving performance in highspeed wide area networks." In *First International Workshop of Protocols for Fast Long-Distance Networks*.

Kephart, J. O. and D. M. Chess (2003): "The vision of autonomic computing." *IEEE Computer*, January.

Keshav, S. (1993): "A control-theoretic approach to flow control." In *Proceedings of the conference on Communications architecture & protocols*, pp. 3–15.

Klein, T. E., K. K. Leung, R. Parkinson, and L. G. Samuel (2004): "Avoiding TCP timeouts in wireless networks by delay injection." In *IEEE Globecom 2004*.

Kleinrock, L. (1970): "A continuum of time-sharing scheduling algorithms." In *AFIPS Conference Proceedings, Spring Joint Computer Conference*, pp. 453–458.

Kleinrock, L. (1975): *Theory, Volume 1, Queuing Systems*. Wiley-Interscience.

Kunniyur, S. and R. Srikant (2002): "A time-scale decomposition approach to adaptive ECN marking." *IEEE Transactions on Automatic Control*, **47:6**, pp. 884–894.

Kwon, W. H. and H. S. Kim (2000): "A survey of control-theoretic approaches in wired and wireless communication networks." In *Proceedings of the Korea-Japan Joint Workshop*.

Lestas, I. and G. Vinniecombe (2004): "Are optimization based Internetcongestion control models fragile with respect to TCP strucutre and symmetry?" In *IEEE Proceedings of Conference on Decission and Control*, pp. 2372–2377. IEEE.

Li, B. and K. Nahrstedt (1999): "A control-based middleware framework for quality of service adaptations." *IEEE Journal on Selected Areas in Communications*, September.

Lin, X. and N. B. Shroff (2004): "Joint rate control and scheduling in multihop wireless networks." In *Proceedings of the IEEE CDC*. Paradise Island, Bahamas.

Lincoln, B. and A. Cervin (2002): "Jitterbug: A tool for analysis of real-time control performance." In *Proceedings of the 41st IEEE Conference on Decision and Control*. Las Vegas, NV.

Liu, S., T. Basar, and R. Srikant (2003a): "Controlling the Internet: A survey and some new results." In *Proceedings of the IEEE Conference on Decision and Control*.

Liu, S., T. Basar, and R. Srikant (2003b): "Controlling the Internet: A survey and some new results." In *Proc. 42nd IEEE Conference on Decision and Control*, pp. 3048–3057. Maui, Hawaii USA.

Loretti, S., P. Soldati, and M. Johansson (2005): "Cross-layer optimization of multi-hop radio networks with multi-user detectors." In *Proceedings of the IEEE WCNC*. New Orleans, LA.

Low, S. H. (2000): “A duality model of TCP and queue management algorithms.” In *Proceedings of ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management*.

Low, S. H. and D. E. Lapsley (1999): “Optimization flow control – I: Basic algorithm and convergence.” *IEEE/ACM Transactions on Networking*, **7:6**, pp. 861–874.

Low, S. H., F. Paganini, and J. C. Doyle (2002a): “Internet congestion control.” *Control Systems Magazine*, **22:1**, pp. 28–43.

Low, S. H., F. Paganini, J. Wang, S. A. Adlakha, and J. C. Doyle (2002b): “Dyanmics of TCP/RED and a scalable control.” In *IEEE Infocom*, vol. 1, pp. 239–248. New York, NY.

Low, S. H. and R. Srikant (2003): “A mathematical framework for designing a low-loss, low-delay Internet.” *Networks and Spatial Economics*, January-February. special issue on "Crossovers between Transportation Planning and Tellecommunications".

Low, S. H. and R. Srikant (2004): “A mathematical framework for designing a low-loss, low-delay Internet.” *Networks and Spatical Economics*, **4**, pp. 75–101.

Lu, C., T. Abdelzaher, J. Stankovic, and S. Son (2001): “A feedback control approach for guaranteeing relative delays in web servers.” In *IEEE Real-Time Technology and Applications Symposium*. TaiPei, Taiwan.

Lu, C., G. A. Alvarez, and J. Wilkes (2002): “Aqueduct: Online data migration with performance guarantees.” In *USENIX Conference on File and Storage Technologies*.

Lu, C., J. Stankovic, T. Abdelzaher, G. Tao, S. Son, and M. Marley (2000): “Performance specifications and metrics for adaptive real-time systems.” In *Proc. 21st IEEE Real-Time Systems Symposium*, pp. 13–23.

Lu, C., J. Stankovic, G. Tao, and S. H. Son (1999): “Design and evaluation of a feedback control EDF scheduling algorithm.” In *Proc. 20th IEEE Real-Time Systems Symposium*, pp. 56–67.

Lu, C., J. A. Stankovic, S. H. Son, and G. Tao (2002): “Feedback control real-time scheduling: framework, modeling and algorithms.” *Real-Time Systems*, **23:1/2**, pp. 85–126.

Lu, C., X. Wang, and C. Gill (2003): “Feedback control real-time scheduling in orb middleware.” In *Proceedings of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium*.

Lu, C., X. Wang, and X. Koutsoukos (2004): “End-to-end utilization control in distributed real-time systems.” In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS), Tokyo, Japan*.

Lu, Y., T. Abdelzaher, C. Lu, L. Sha, and X. Liu (2003): “Feedback control with queueing-theoretic prediction for relative delay.” In *IEEE Real-Time and Embedded Technology and Applications Symposium*.

Lu, Y., T. Abdelzaher, C. Lu, and G. Tao (2002): “An adaptive control framework and its application to differentiated caching services.” In *International Conference on Quality of Service*. Miami Beach, FL.

Lu, Y., A. Sexana, and T. Abdelzaher (2001): "Differentiated caching services; a control-thoeretical approach." In *Proceedings of the 2001 International Conference on Distributed Computing Systems*, pp. 615–622.

Ludwig, R. and R. H. Katz (2000): "The Eifel algorithm: Making TCP robust against spurious retransmissions." *ACM Computer Communication Review*, **30:1**.

Madan, R., S. Cui, S. Lall, and A. Goldsmith (2005): "Cross-layer design for lifetime maximization in interference-limited wireless sensor networks." In *Proceedings of the IEEE Infocom*. Miami, FL. To Appear.

Mascolo, S., C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang (2001): "TCP Westwood: bandwidth estimation for enhanced transport over wireless links." In *MobiCom*. Rome, Italy.

Massoulie, L. (2000): "Stability of distributed congestion control with heterogeneous feedback delays." Technical Report. Microsoft Research, Cambridge, UK.

Mathis, M., J. Mahdavi, S. Floyd, and A. Romanow (1996): "TCP selective acknowledgements options." RFC 2018.

Mazenc, F. and S. I. Niculescu (2003): "Remarks on the stability of a class of TCP-like congestion control models." In *IEEE Proceedings of Conference on Decission and Control*, pp. 5591–5594. IEEE.

Möller, N. and K. H. Johansson (2003): "Influence of power control and link-level retransmissions on wireless TCP." In *Quality of Future Internet Services*, vol. 2811 of *Lecture Notes in Computer Science*. Springer-Verlag.

Möller, N., K. H. Johansson, and H. Hjalmarsson (2004): "Making retransmission delays in wireless links friendlier to TCP." In *Proc. 43rd IEEE Conference on Decision and Control*. IEEE.

Nahrstedt, K. (1995): "End-to-end qos guarantees in networked multimedia system." *ACM Computing Surveys*, **27:4**.

Nakajima, T. (1998): "Resource reservation for adaptive QoS mapping in real-time Mach." In *Proceedings of Sixth International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS)*.

Neely, M. J., E. Modiano, and C. E. Rohrs (2003): "Dynamic power allocation and routing for time varying wireless networks." In *Proceedings of the IEEE Infocom*. San Francisco, CA.

Niculescu, S. I. (2001): *Delay Effects on Stability*. Springer.

Paganini, F., J. C. Doyle, and S. H. Low (2001): "Scalable laws for stable network control." In *IEEE CDC*, vol. 1, pp. 185–190. Orlando, FL.

Palopoli, L., L. Abeni, and G. Lipari (2003): "On the application of hybrid control to CPU reservations." In *Proceedings of the Conference on Hybrid Systems Computation and Control (HSCC03)*.

Papachristodoulou, A., L. Li, and J. C. Doyle (2004): "Methodological frameworks for large-scale network analysis and design." *ACM SIGCOMM Computer Communications Review*, **34:3**, pp. 7–20.

Peet, M. and S. Lall (2004): "On global stability of Internet congestion control." In *IEEE Proceedings of Conference on Decission and Control*, pp. 1035–1041. IEEE.

Pitsillides, A., J. Lambert, and D. Tipper (1995): "A multilevel optimal control approach to dynamic bandwidth allocation in broadband ISDN." *Telecommunication Systems*, **4**, pp. 71–96.

Potier, D., E. Gelenbe, and J. Lenfant (1976): "Adaptive allocation of central processing unit quanta." *Journal of the ACM*, **23:1**, pp. 97–102.

Radunovic, B. and J.-Y. L. Boudec (2004): "Rate performance objectives of multihop wireless networks." In *Proceedings of the IEEE Infocom*. Hong Kong.

Robertazzi, T. G. (1994): *Computer Networks and Systems: Queuing Theory and Performance Evaluation*. Springer-Verlag.

Robertsson, A., B. Wittenmark, and M. Kihl (2003): "Analysis and design of admission control in web-server systems." In *Proceedings of ACC'03*.

Romn, M., C. Hess, R. Cergueira, R. Campbell, and K. Nahrstedt (2002): "Gaia: A middleware infrastructure to enable active spaces." *IEEE Pervasive Computing*, Oct-Dec.

Samaraweera, N. K. G. (1999): "Non-congestion packet loss detection for TCP error recovery using wireless links." *IEEE Proceedings-Communications*, **146:4**, pp. 222–230.

Sarolahti, P., M. Kojo, and K. Raatikainen (2003): "F-RTO: an enhanced recovery algorithm for TCP retransmission timeouts." *ACM SIGCOMM Computer Communication Review*, **33:2**.

Seto, D., J. P. Lehoczky, and L. Sha (1998): "Task period selection and schedulability in real-time systems." In *Proc. 19th IEEE Real-Time Systems Symposium*, pp. 188–198. Madrid, Spain.

Sha, L. (1998): "Dependable system upgrade." In *Proc. IEEE Real Time Systems Symposium*.

Sha, L. (2001): "Using simplicty to control complexity." *IEEE Software*, **18:4**.

Sha, L., X. Liu, Y. Lu, and T. Abdelzaher (2002): "Queuing model based network server performance control." In *IEEE Real-Time Systems Symposium*.

Sharma, S. and D. Tipper (1993): "Approximate models for the study of nonstationary queues and their applications to communication networks." In *Proc. of IEEE International Conference on Communications*, pp. 352–358.

Skadron, K., T. Abdelzaher, and M. R. Stan (2001): "Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management." In *International Symposium on High Performance Computer Architecture*.

Speranzon, A. and K. Johansson (2003): "Distributed pursuit-evasion game: Evaluation of some communication schemes." In *The Second Annual Symposium on Autonomous Intelligent Networks and Systems*.

Srikant, R. (2004): *The Mathematics of Internet Congestion Control*. Birkhauser.

Stankovic, J. A., C. Lu, S. H. Son, and G. Tao (1999): "The case for feedback control real-time scheduling." In *Proc. 11th Euromicro Conference on Real-Time Systems*, pp. 11–20.

Stevens, W. (1997): "TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms." RFC 2001.

Tipper, D. and M. K. Sundareshan (1990): "Numerical models for modeling computer networks under nonstationary conditions." *IEEE Journal on Selected Areas in Communications*, **8:9**, pp. 1682–1695.

Toumpis, S. and A. Goldsmith (2002): "Capacity regions for wireless ad hoc networks." In *Proceedings of the IEEE International Conference on Communications*. New York, NY.

Värbrand, P., D. Yuan, and P. Björklund (2003): "Resource optimization of spatial TDMA in ad hoc radio networks: a column generation approach." In *Proceedings of the IEEE Infocom*. San Francisco, CA.

Vinnicombe, G. (2002): "On the stability of networks operating TCP-like congestion control." In *IFAC World Congress*. Barcelona, Spain.

Vinnivombe, G. (2000): "On the stability of end-to-end congestion control for the Internet." Technical Report CUED/F-INFENG/TR.398. Cambridge University, Cambridge, UK.

Wang, W., D. Tipper, and S. Banerjee (1996): "A simple approximation for modeling nonstationary queues." In *Proc. of IEEE Infocom'96*, pp. 255–262.

Wang, X., H.-M. Huang, V. Subramonian, C. Lu, and C. Gill (2004): "Camrit: Control-based adaptive middleware for real-time image transmission." In *Proceedings of IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS) Toronto, Canada*.

Wen, J. T. and M. Arcak (2004): "A unifying passivity framework for network flow control." *IEEE Transactions on Automatic Control*, **49:2**, pp. 162–174.

Xiao, L., M. Johansson, and S. Boyd (2002): "Simultaneous routing and resource allocation via dual decomposition." In *Proceedings of the 4th Asian Control Conference*, pp. 29–34. Singapore.

Yavatkar, R. and N. Bhagawat (1994): "Improving end-to-end performance of TCP over mobile internetworks." In *Workshop on Mobile Computing Systems and Applications*.

Yi, Y. and S. Shakkottai (2004): "A hop-by-hop congestion control over a wireless multi-hop network." In *Proceedings of the IEEE Infocom*.

Ying, L., G. E. Dullerud, and R. Srikant (2004): "Global stability of internet congestion controllers with heterogeneous delays." In *Proceedings of the American Control Conference 2004*.

Zhang, Q., W. Zhu, and Y.-Q. Zhang (2001): "Resource allocation for multimedia streaming over the internet." *IEEE Transactions on Multimedia*, **3:3**, pp. 339–355.

Zhang, R., C. Lu, T. F. Abdelzaher, and J. A. Stankovic (2002): "Controlware: A middleware architecture for feedback control of software performance." In *Proceedings of the 2002 International Conference on Distributed Computing Systems*. Vienna, Austria.