Networked Control Systems

Richard M. Murray
Control and Dynamical Systems
California Institute of Technology

DRAFT v1.0a, 10 March 2008

Chapter 1
Introduction

Modern control theory is largely based on the abstraction that information ("signals") are transmitted along perfect communication channels and that computation is either instantaneous (continuous time) or periodic (discrete time). This abstraction has served the field well for 50 years and has led to many success stories in a wide variety of applications.

Future applications of control will be much more information-rich than those of the past and will involve networked communications, distributed computing, and higher levels of logic and decision-making (see [Mur03] for a recent analysis of future directions in this area). New theory, algorithms, and demonstrations must be developed in which the basic input/output signals are data packets that may arrive at variable times, not necessarily in order, and sometimes not at all. Networks between sensors, actuation, and computation must be taken into account, and algorithms must address the tradeoff between accuracy and computation time. Progress will require significantly more interaction between information theory, computer science, and control than ever before.

An emerging framework for networked control systems is shown in Figure 1.1. This architecture separates the traditional elements of sensing, estimation, control, and actuation for a given system across a network and also allows sharing of information between systems. As we will see in the examples below, careful decisions need to be made on how the individual components in this architecture are implemented and how the communications across the networked elements is managed. This architecture can be used to model either a single system (using either half of the diagram) or multiple systems that interact through the network.

One example of the use of this architecture is autonomous operations for sensor-rich systems, such as unmanned, autonomous vehicles. As part of the 2004 and 2005 DARPA Grand Challenges, Caltech has developed two such vehicles ("Bob" and "Alice") that each make use of a networked control systems architecture. Alice, the 2005 vehicle, has six cameras, 4 LADAR units, an inertial meaurement unit (IMU), a GPS navigation system, and numerous internal temperature and vibration sensors. The raw data rate for Alice is approximately 1–3 Gb/s, which must be processed and acted upon at rates of up to 100 Hz in order to insure safe operation at high driving speeds.

The control system for Alice makes use of the architecture depicted in

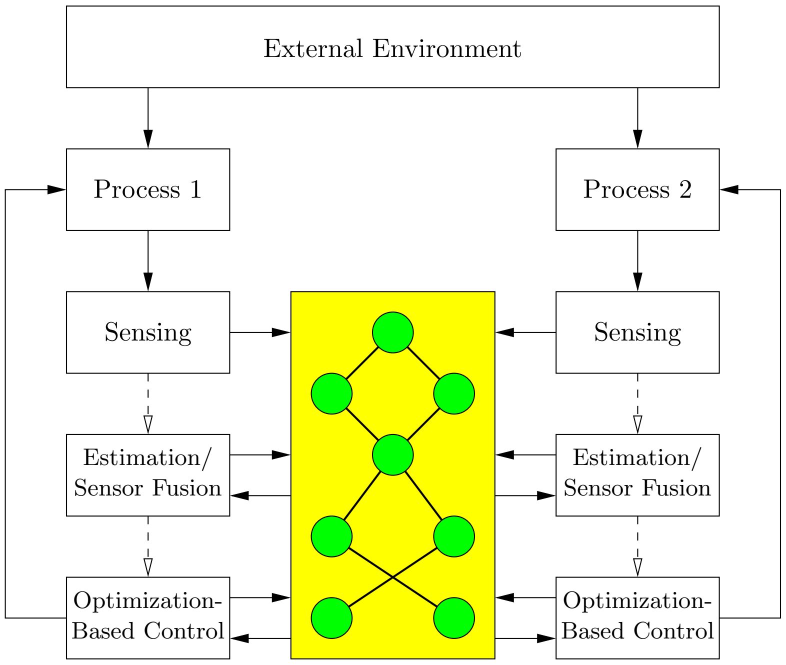
External Environment
Process 1
Process 2
Sensing
Sensing
Estimation/ Sensor Fusion
Estimation/ Sensor Fusion
Optimization-Based Control
Optimization-Based Control

Figure 1.1: Emerging framework for networked control systems. Signals between control system modules for multiple processes are transmitted through a communication network.

Figure 1.1, with distributed data fusion to determine elevation maps (for the height of the terrain in front of the vehicle), multiple optimization-based controllers to plan possible routes for the vehicle, and online modeling, fault management, and decision making to provide reliable and reconfigurable operation. Eight onboard computers distribute the computational load, sharing information at mixed rates across a 1 Gb/s switched network. System specifications call for reliable operation in the presence of up to 1 computer failure and 2 sensor failures, requiring careful coordination between computational elements.

A major challenge in Alice is determining how to send information between nodes. Because of the high data rates and computational loads on the CPUs, packets sent across the network are not always received and the system must be robust to various networking effects. The choice of protocols and design of the overall messaging system is currently informal and based on trial and error. As an example of the issues that must be resolved, certain packets of data are very important, such as packets containing raw sensor information from a portion of the terrain that is scanned only once. Other data can be dropped if needed, such as commanded trajectories (the old trajectory can be used for several sampling periods). Data from the inertial measurement unit must be received with minimum latency, while other data (a change in the temperature of the vehicle) is much less time

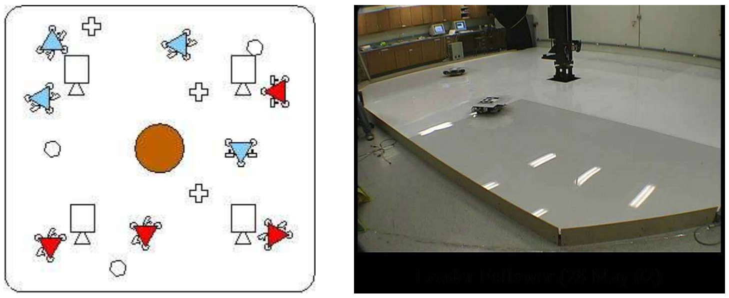

Figure 1.2: The Caltech Multi-Vehicle Wireless Testbed. The left figure shows the layout of the testbed area, including overhead cameras and fixed communication nodes (crosses and hexagons). The right picture is the current laboratory, with two vehicles shown.

critical. Substantial effort has been put into trying to make sure that the computations and network protocols complement each other and that loss of data and data latency does not degrade the performance of the system.

Another example of a networked control system is illustrated by the Caltech Multi-Vehicle Wireless Testbed (MVWT, shown in Figure 1.2), which consists of a collection of 8-12 vehicles performing cooperative tasks. The MVWT represents a slightly different instantiation of the architecture in Figure 1.1: each vehicle has a single processor with full access to local sensing and actuation, but information between vehicles must be sent across the network. The wireless commmuncation channels can exhibit significant degredation when multiple vehicles are attempting to communication and packet loss rates of 5-15% are not uncommon.

The issues in desiging a cooperative control policy for the MVWT vehicles faces many of the same challenges as those seen in Alice. Information communicated between vehicles can be dropped, reordered or sent with variable delay. Sensor information required for overall situational awareness can be fused at multiple levels and/or in a distributed fashion. Again, the currently available protocols for network communications are not well tuned to operation in this type of environment. For example, bit errors in packets can result in losing the entire data packet, rather than passing the information to the applications layer where partial (lossy) information could still be used effectively.

A more detailed architecture for a networked control system is shown in Figure 1.3. At the top of the figure, the standard elements for a control system are present: actuation, system dynamics, sensing and environmental disturbances and noise. For many networked control systems, the amount of sensory information available is very large, requiring care in how this information is transmitted. Alice, for example, had between 1 and 3 gigabits/second (Gb/s) raw data rate, depending on the sensor suite taht was used. Another difference with traditional control systems is that the actua-

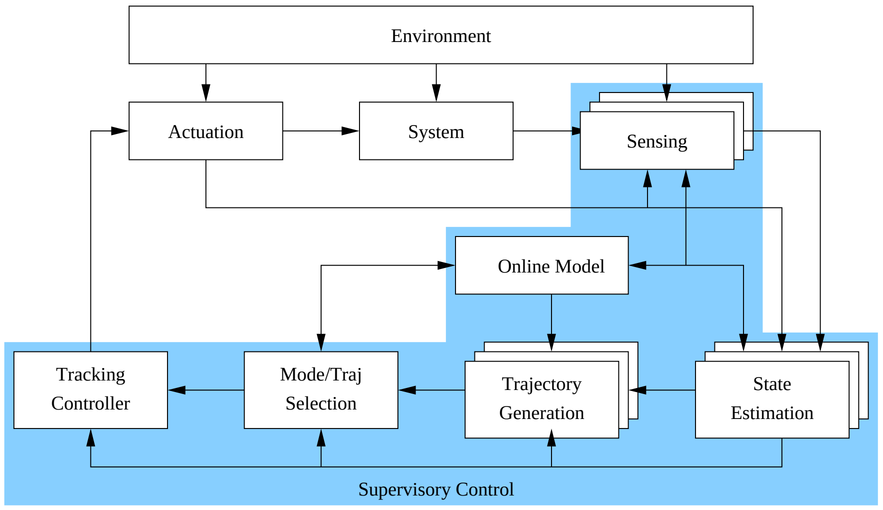

Figure 1.3: A detailed architecture for a networked control system, based on the control system for Alice [**?**].

tion subsystems are themselves embedded systems, capable of some amount of computation and local memory storage.

The primary control loop in a networked control system consists of state estimation, trajectory generation, and trajectory tracking. These elements can all represent relatively substantial computations (depending on the application) and are linked to each other through a number of network ports. In Alice, for example, the state estimation modules included a traditional inertial state estimator (combining GPS data with gryos and accelerometer measurements) as well as four computers that were estimating terrain information and computing a fused "speed map" that described the maximum allowable velocity that could be used in a given area of the terrain in front of it (more details on the software for Alice is given in Appendix **??**).

The information from the state estimators is used by trajectory generation algorithms that compute the desired state and inputs for the system to accomplish a task or minimize a cost function. The trajectory generation algorithms are responsible for taking into account actuator and state constraints on the sytem, as well as the nonlinear nature of the underlying process dynamics. A typical approach for these algorithms is to perform optimization-based control, in which one attempts to minimize a cost function subject to satisfying the constraints and dynamics. With the advances in computational power, it is often possible to run these optimization-based planners quickly enough that they can recompute the path from the current location in a "receding horizon" fashion, allowing feedback at the planner level. This is particularly useful to manage uncertainty in the cost function, for example when the cost is determined in real-time (as in the case of Alice, where the cost is based no the terrain that is being traversed).

As in the case of state estimation, networked control systems often use more than one trajectory generation algorithm running simultaneously. Since the physical system can only track one trajectory, some level of mode management and trajectory selection is required. This mode or trajectory selection logic is often under the control of higher levels of decision making (supervisory control).

The last element of the primary control loop is the trajectory tracking module, which is responsible for high frequency disturbance rejection and tracking. This module is itself a feedback system, using the state estimate and the desired trajectory to compute the actuation commands. In the context of a networked control system, the primary difference with traditional trajectory tracking algorithms is the need to run in a asynchronous execution environment, where reference trajectories and sensory measurements may come in at varying rates, including short periods where no inputs may arrive (due to network delays, computational delays or fault handling in one of the other modules).

In addition to the elements of the primary control loop, networked control systems can also contain a number of modules responsible for higher levels of decision making. We loosely refer to these modules as "supervisory control": they are responsible for implementing various control system functions that involve choosing parameters used by the primary control loop (such as cost functions and communication rates), dealing with failures of hardware and software components, maintaining an online model of the system dynamics, and adapting the performance of the system based on observed behaviors and memory. While these elements are critical for the operation of a networked control systems, in this text we focus on the primary control loop, where the network effects are most directly relevant.