Fundamental issues in networked control systems – 3: Architecture and Theory

P. R. Kumar

Dept. of Electrical and Computer Engineering, and Coordinated Science Lab, University of Illinois, Urbana-Champaign

Email: prkumar@illinois.edu
Web: http://decision.csl.illinois.edu/~prkumar
Networked Cyber-Physical Systems

Wireless networks

Sensor Networks

Networked Embedded Control
The Third Generation of Control Systems

- **First generation: Analog Control Systems**
  - Technology: Electronic Feedback Amplifiers
  - Theory: Frequency domain analysis: Bode, Evans

- **Second Generation: Digital Control**
  - Technology: Digital computers
  - Theory: State-space design, Kalman filter, $H_\infty$, Real-Time Scheduling

- **Third generation: Networked Embedded Control Systems**
  - Embedded computers
  - Wireless and wireline networking
  - Software
Intertwined histories of communication, computation and control

- Perhaps the most exciting developments in the information area relate to the large-scale digital computing machines.”
  – Claude Shannon, 1947

- “I think I can claim credit for transferring the whole theory of the servomechanism bodily to communication engineering.”
  – Norbert Wiener, 1956

- “…the era of cyberspace and the Internet, with its emphasis on the computer as a communications device and as a vehicle for human interaction connects to a longer history of control systems that generated computers as networked communications devices.”
  – David Mindell in “Feedback, Control and Computing before Cybernetics,” 2002
The themes

- Time
- Information
- Abstractions
- Analysis
- Applications
Challenge of Abstractions and Architecture

What are the abstractions and architecture for convergence with communication and computing?

Goal is to enable rapid design and deployment
- Critical Resource: Control Designer’s Time

Standardized abstractions and architecture
- Minimal reconfiguration and reprogramming

Hopefully leading to massive proliferation
Architecture and computing, communication and control

- From general purpose Computing
  - Building software systems on a standard platform
    - Standard abstractions and architecture
    - Program as data
    - Interfaces
    - Ease of customization
    - Modular design of software
    - Portability – High level languages
    - Reusability – Component libraries
  
- To general purpose Networking
  - Building computer networks around a standard architecture
    - Standard software architecture
    - TCP/IP is de facto standard
    - Layers of protocols
    - Breakdown networking into sub-problems
    - Solve sub-problems in different layers
    - Compose solutions into a working stack
    - Mechanism: Encapsulation of packets

- From general purpose Computing
  - IBM 360: General purpose computer
    - First commercially successful model
    - 360 – All round capability

- To general purpose systems
  - Building operational control applications with generic components in a standard framework
  - Example: PID Controller
    - Generic tunable component
    - Easy to integrate
    - Key concept: Standard interfaces

What should be the architecture of third generation systems?
Information Technology Convergence Lab:
The Systems

(Baliga, Graham, Huang & K '02)
Abstraction of Virtual Collocation (Graham, Baliga & K ‘05)
The Abstraction Layers

Application Layer
- System Layer
- Transport Layer
- Network Layer
- Link Layer

(Graham, Baliga & K '05)
The Abstraction Layers

Application Layer

System Layer

Transport Layer

Network Layer

Link Layer

(Graham, Baliga & K ‘05)
The Abstraction Layers

- Middleware manages the Components

<table>
<thead>
<tr>
<th>Application Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Layer</td>
</tr>
<tr>
<td>Transport Layer</td>
</tr>
<tr>
<td>Network Layer</td>
</tr>
<tr>
<td>Link Layer</td>
</tr>
</tbody>
</table>
The Abstraction Layers

Middleware manages the Components

- Minimal reconfiguration and reprogramming

Etherware
- Location independence
- Semantic addressing of components
- System startup and upgrade during execution
- Automatic migration of components for performance

(Baliga, Graham, & K '05)
Collision avoidance
(Schuetz, Robinson & K ‘05)

http://decision.csl.uiuc.edu/~testbed/videos/CollisionAvoidance.mpg
Local Temporal Autonomy

- Components able to tolerate failures of other components for some time

- **Example**: Insulating Controller from Sensor and Communication Network

  ![Diagram](sensor-controller-diagram.png)

  - Sensor → State Estimator → Controller
  - Now controller has Local Temporal Autonomy

- **Example**: Insulating Actuator from Controller and Communication Network

  ![Diagram](controller-actuator-diagram.png)

  - Controller → Block Computation → Actuator Buffer → Actuator
  - Now Actuator has Local Temporal Autonomy

- Converts Dependency relationships to Use If Available relationships

- Makes possible other facilities such as
  - Automatic Restart of Failed Components
  - Migration of Components
  - Component Upgrade at Runtime

  Reliability, robustness
  System integration, Initialization
  Evolution and Scalability
Example of capabilities: Component Migration

- Designer should not have to deal with such low level issues
  - Designer’s time is the critical resource

Communicate pixels? Or compute position?

Kalman filter

Car controller

Computer 1

Computer 2

Excessive communication overhead

Migrate Kalman Filter to Computer 2:
Done through Memento Design pattern
Component Migration at Run-Time

- Migrate a controller to another location while system is running

http://decision.csl.uiuc.edu/~testbed/videos/migration.mpg
Forcing functions and application design: Delays and Losses

Forcing function
- Wireless channels have unpredictable delays and high losses

Application design constraint
- Use state estimators and control buffers

Etherware design
- Assume components can tolerate some delays (no hard deadlines)

Benefits
- Application can tolerate some delays – soft real time
- Can use general purpose platforms (OS, languages, etc)
Robustness

- Components must have high availability

Application design constraint
- Check-point component state for quick restarts

Etherware design
- Components implement an explicit state check-pointing interface

Benefits
- Initialization can be performed using default initial state
- Checkpoints can be used for component upgrades and migration
Five themes

- How much traffic can wireless networks carry?
- How should networks compute functions of distributed data?
- How can clocks be synchronized over networks?
- What are the appropriate abstractions and architecture?
- Where to locate control laws?
- How to establish properties of overall hybrid systems?
Challenge of locating and optimizing control laws

- Many open problems
- Optimizing with respect to delays, failures, structures, etc
  - Migration, Reconfiguration, etc
- What are the appropriate theories to take advantage of the next generation capabilities
- Currently bottleneck is Theory, not Technology
Location and optimization of control over a lossy network

- LQG Problem:
  \[ x_{k+1} = Ax_k + Bu_k + Gw_k \]
  \[ y_k = Cx_k + Hv_k, \]
  \[ \text{Min } E \left[ x_N^tFx_N + \sum_{k=0}^{N-1} \left( x_k^tQx_k + u_k^tRu_k \right) \right] \]
  Non-degeneracy assumptions

- Lossy network
  - IID Packet drops

- What is the best location for controller that calculates control law?

- And what should that control law be?

- Non-classical information pattern
  - Memory of past actions is lost

- Witsenhausen (1968)
  - Computation of optimal controller is intractable
The Long Packet Assumption (LPA) (Robinson & K ‘06)

◆ **Theorem**
  - Suppose that “long packets” containing all history are transmitted
  - Then optimal location for Controller is at Actuator
  - The optimal control law is:
    \[ u_k^* = - \left[ R + B' W_{k+1} B \right]^{-1} B' W_{k+1} A \hat{x}_{k|k-D-1} \]
  - Provides a lower bound on cost without long packets
  - Even without LPA, optimal controller can be calculated when control is at actuator (Sinopoli, Schenato, Franceschetti et al ‘04, Imer et al ‘06)
    - No Witsenhausen problem when controller is placed at actuator
    - Can compare this suboptimal controller with lower bound
The Long Packet Assumption (LPA)

- **Theorem (Robinson & K ’06)**
  - Suppose that “long packets” containing all history are transmitted
  - Then optimal location for Controller is at Actuator
  - \( \exists \) policy with finite cost and \( \lim_{k \to \infty} \frac{E[x'_k x_k]}{k} = 0 \) iff \( |\lambda_{\max}(A)| < \frac{1}{\max(eig(A))^2} \)
  - The optimal long-term average quadratic cost per is:
    \[
    \sum_{D=0}^\infty \lambda^D (1-\lambda) \left\{ (1 - \lambda) Tr \left( A' \left( A^{D+1} W A^{D+1} - W \right) A \Sigma_0 \right) + \sum_{i=1}^{D+1} A^i W A^i \Sigma^w \right\} + Tr \left( \Sigma_{D+1}^w Q \right) \]
    It is a lower bound on cost without long packets.
  - Optimal controller is \( u^*_k = - [R + B' W B]^{-1} B' W A \hat{x}_{k|k-D_k-1} \) Slight strengthening of separation theorem

- Even without LPA, optimal controller can be calculated when control is at actuator (Sinopoli, Schenato, Franceschetti et al ‘04, Imer et al ‘06)
  - No Witsenhausen problem when controller is placed at actuator
  - Can compare this suboptimal controller with lower bound
Near optimality of placing controller at actuator

Problem is effectively solved

\[ A = \begin{bmatrix} 1.7000 & 0.0000 & 0.0000 & 0.0000 \\ 0.6211 & 0.6029 & 0.3028 & 0.6965 \\ 0.3200 & 0.7210 & 0.3685 & 0.1146 \\ 0.2272 & 0.7875 & 0.0138 & 0.2518 \end{bmatrix} \]
Five themes

- How much traffic can wireless networks carry?
- How should networks compute functions of distributed data?
- How can clocks be synchronized over networks?
- What are the appropriate abstractions and architecture?
- Where to locate control laws?
- How to establish properties of overall hybrid systems?
The themes

- Time
- Information
- Abstractions
- Analysis
- Applications
Automatic traffic control: Safety and Liveness guarantees

- Hybrid System:
  - Bridging between discrete and continuous models

(Giridhar, Graham, Baliga & K ‘03)

http://decision.csl.uiuc.edu/~testbed/videos/city_7cars.mpg
Analyzing the whole CPS system: System Safety and Liveness

**Theorem** (Baliga & K ’05)

- Directed graph model of road network
  - Each bin has in-degree 1 or out-degree 1
  - System has no occupied cycles initially

\[ W = R(1 - \cos \beta(2 \cos \alpha - 1)) \]

- Road width:
  \[ (d, \theta) : d + R(1 - \cos \theta) < W \]

  - Initial condition: \[ \leq \gamma \]
  - Intersection angles, and road lengths:
    - Multiple cars with appropriate spacing

- Car control model: Kinematic model with turn radii \( R \) and

- Real time renewal tasks: HST scheduling with

- Then cars can be operated
  - Without collisions (Safety) or
  - Gridlocks (Liveness)

http://decision.csl.uiuc.edu/~testbed/videos/city_7cars.mpg
The themes

- Time
- Information
- Abstractions
- Analysis
- Applications
An application: Intelligent Intersections

- Stop signs and Traffic lights are wasteful
  - At suburban or rural intersections
  - At night, when traffic load is very low

- Goal: Intelligent intersections where vehicles coordinate their movement through a combination of centralized and distributed real-time decision-making
  - Lower fuel consumption
  - Lower traffic delays
  - Greater throughput

- Assign time-slots to each car and design distributed algorithm which ensures safety

- “Electronic equivalent of a Traffic light”
Intelligent Intersections

- Stop signs & Traffic lights wasteful
  - Suburban or rural intersections
  - At night, when traffic light

- Goal: *Intelligent* intersections
- Cars negotiate with intersection
  - Lower fuel consumption
  - Lower traffic delays
  - Greater safety

(Kowshik, Caveney & K '05)
But are we solving the right problem?

- Human beings seem to have decided to dedicate one hour per day for travel.
- So should we actually make it easier for people to travel greater distances in the same time?

TRAVEL TIME BUDGET: GLOBAL DATA


From Frank Kelly’s talk at StochNet 2006
The oncoming theoretical convergence

Post Maxwell, von Neumann, Shannon, Bardeen-Brattain world

Age of system building
Nodes can Compute
Communicate
Sense and Actuate

- 1950 — 2000 and continuing: Substantial progress in several individual disciplines
  - Computation: ENIAC (1946), von Neumann (1944), Turing,..
  - Sensing and inference: Fisher, Wiener (1949),…
  - Actuation/Control: Bode, Kalman (1960),…
  - Communication: Shannon (1948), Nyquist,…
  - Signal Processing: FFT, Cooley-Tukey (1965),…

- Larger grand unification of sensing, actuation, communication and computation

- 2000 — onwards
  - A gradual fusion of all these fields
  - But still knowledge of all these fields may be important
  - Pedagogical as well as research challenges
References-1


References-3


References-4

References-5


References-6

Thank you
Attribution-Noncommercial-No Derivative Works 3.0 Unported

You are free:

- to Share — to copy, distribute and transmit the work.

Under the following conditions:

- Attribution — You must attribute this work to P. R. Kumar (with link).
  
  Attribution:
  
  http://decision.csl.illinois.edu/~prkumar/talks.html

- Noncommercial — You may not use this work for commercial purposes.

- No Derivative Works — You may not alter, transform, or build upon this work.

With the understanding that:

- Waiver — Any of the above conditions can be waived if you get permission from the copyright holder.

- Other Rights — In no way are any of the following rights affected by the license:
  
  - Your fair dealing or fair use rights;
  - The author's moral rights;
  - Rights other persons may have either in the work itself or in how the work is used, such as publicity or privacy rights.

- Notice — For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to this web page.

This is a human-readable summary of the Legal Code (the full license).