ESD Electronic Systems Design

Synthesis Verification Testing Power Communication

Design of Networked Control System

Davide Quaglia,

Riccardo Muradore

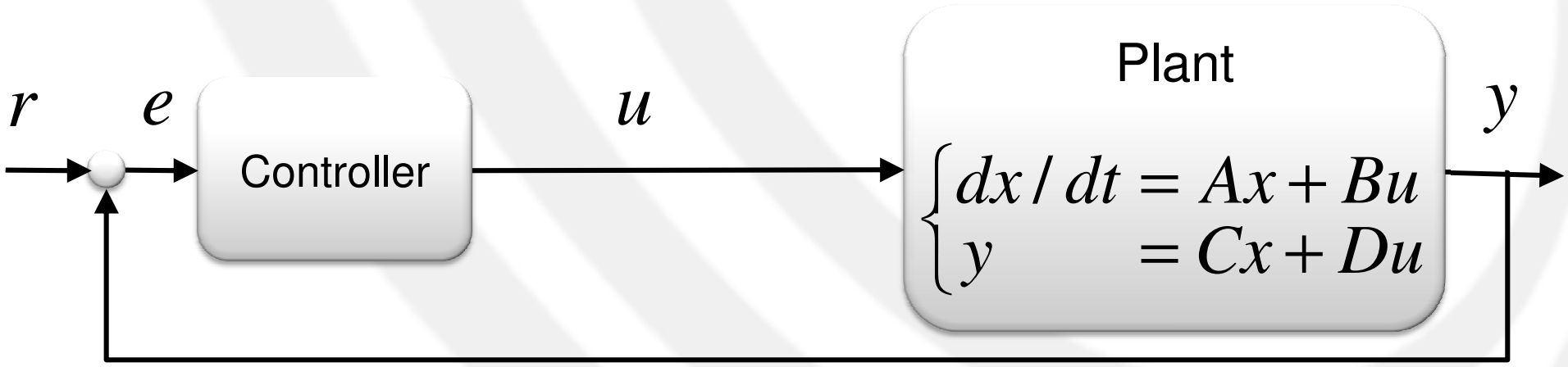
Outline

- Definition
- Applications
- Control problems…
- … and some design solutions
  - Differentiated services
  - Transmission power adaptation
- Simulation problems
- Relationship between NES and NCS

DEFINITION

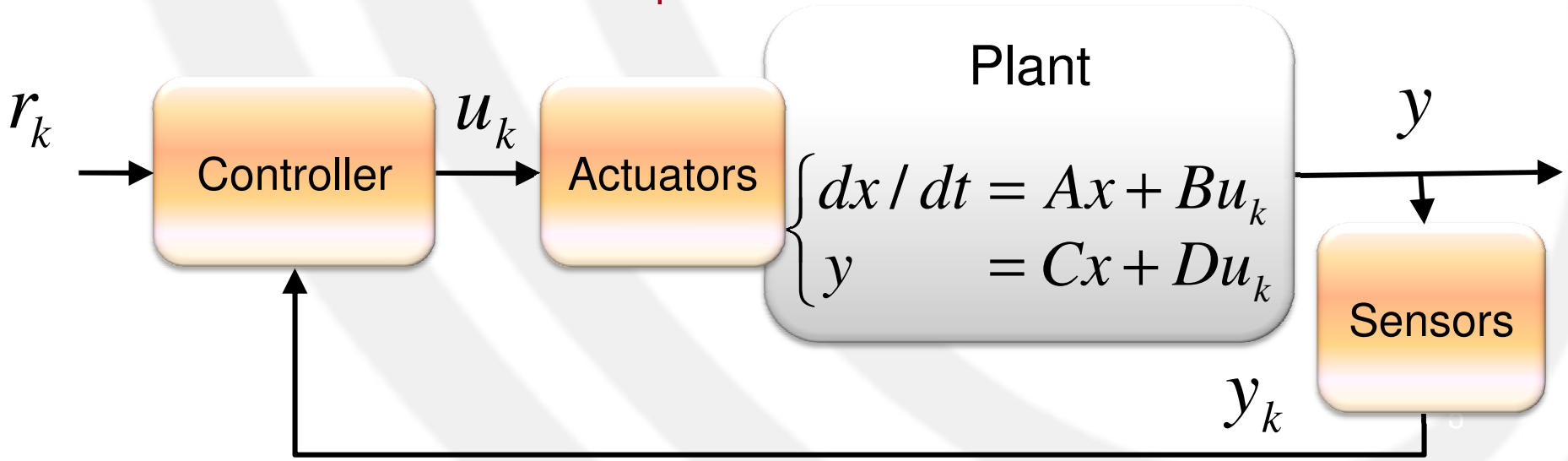
Automation control system

- According to Control Theory, an Automation Control System consists of
  - System to be controlled (Plant)
  - Controller



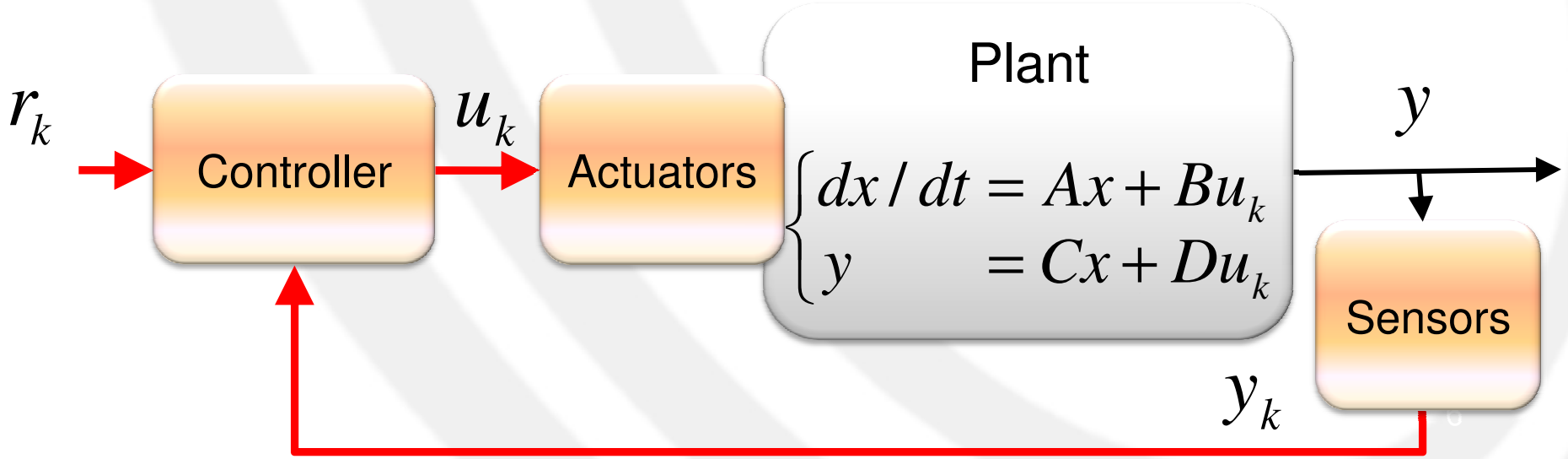
Architecture of a control system

- The Plant is the **given** physical system
- The blocks built by the designer are:
  - Controller = SW program implementing the control strategy (it computes also e_k)
  - Sensors to read information from the plant
  - Actuators to act on the plant



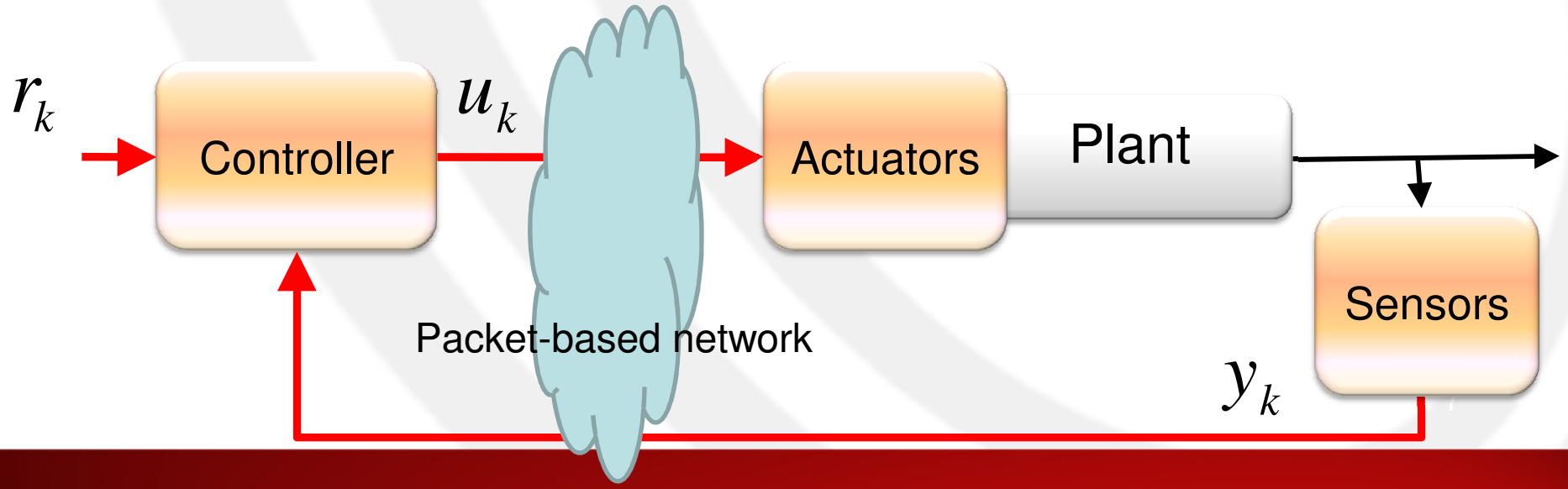
Cyber-Physical System

- Controller, actuators and sensors are digital blocks evolving according to discrete events (cyber)
- The plant follows physical (continuous time) laws
- The red connections move digital information

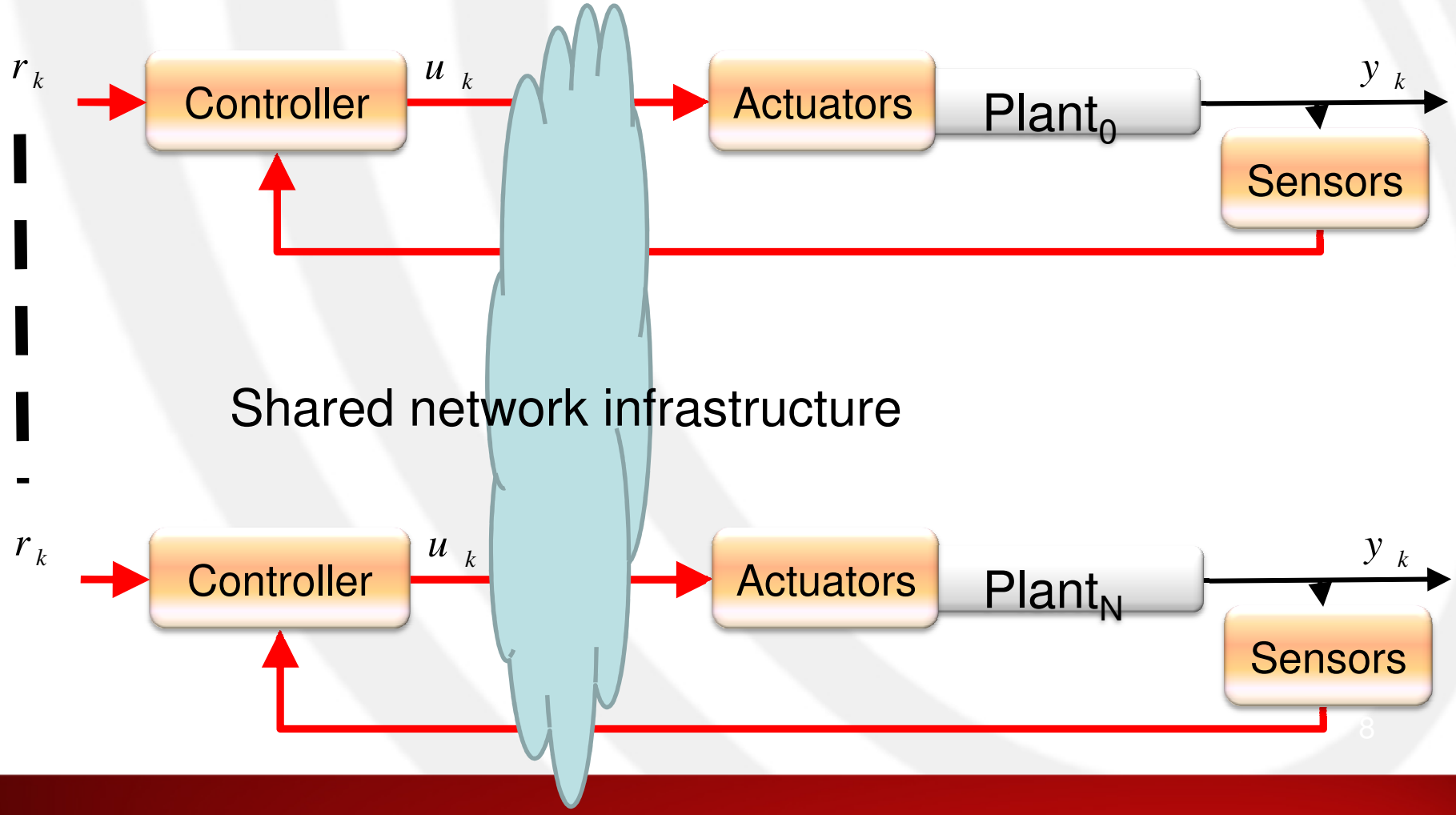


Networked Control System

- A packet-based network delivers both commands (forward) and measurements (backward)
  - Scalability
  - Cost efficiency



Scalability



Cost efficiency

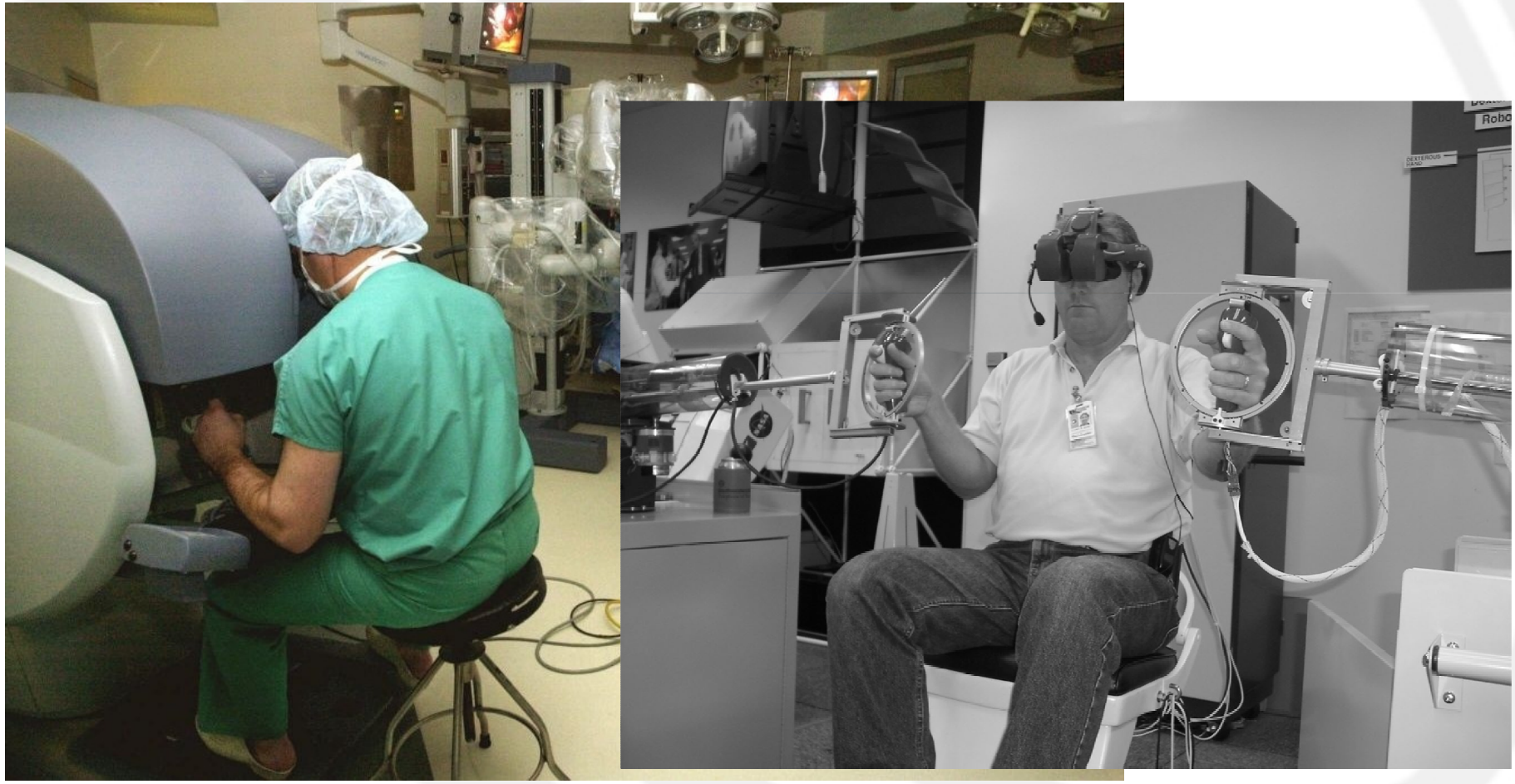
- Single network infrastructure for the growing needs of the factory
- Well-known re-used protocols and equipments
  - Ethernet
  - CAN
  - Wifi (incredible !)
  - TCP/IP

APPLICATIONS

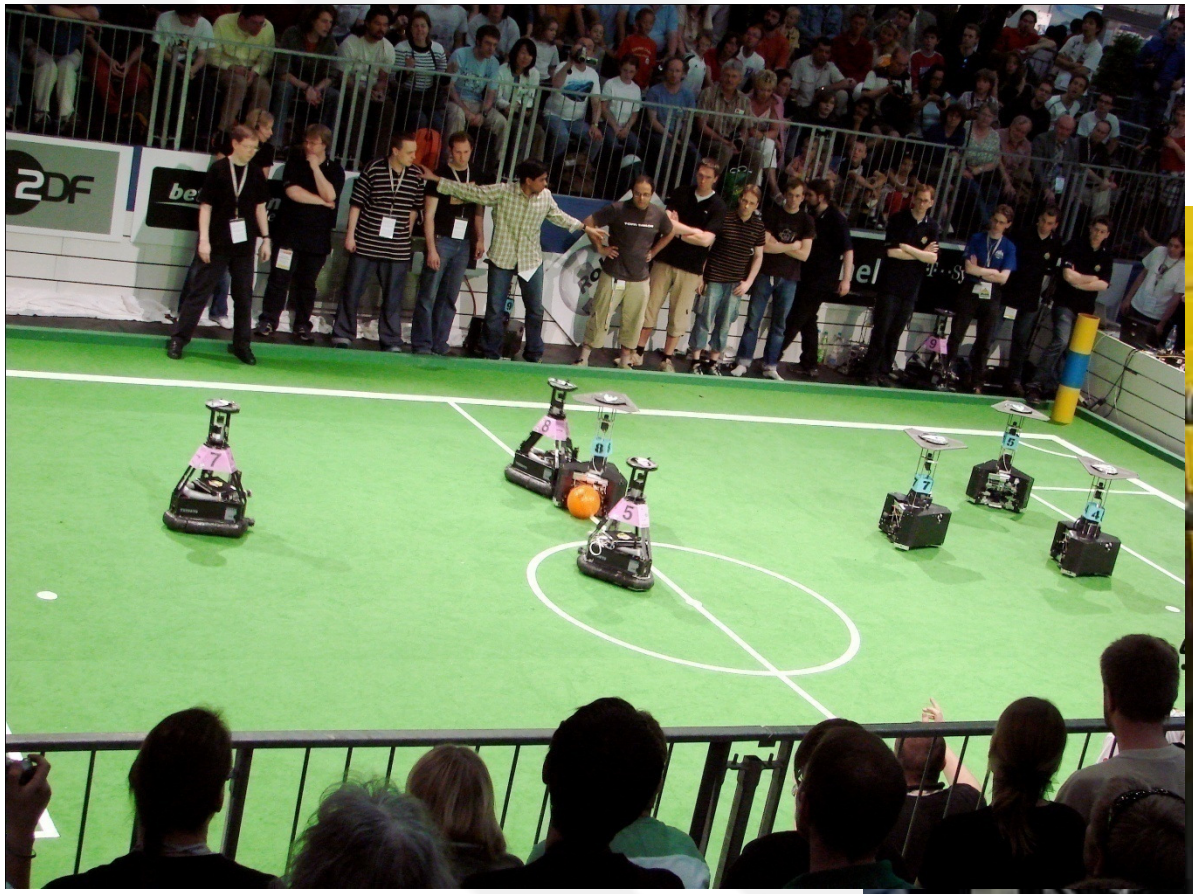
Space exploration

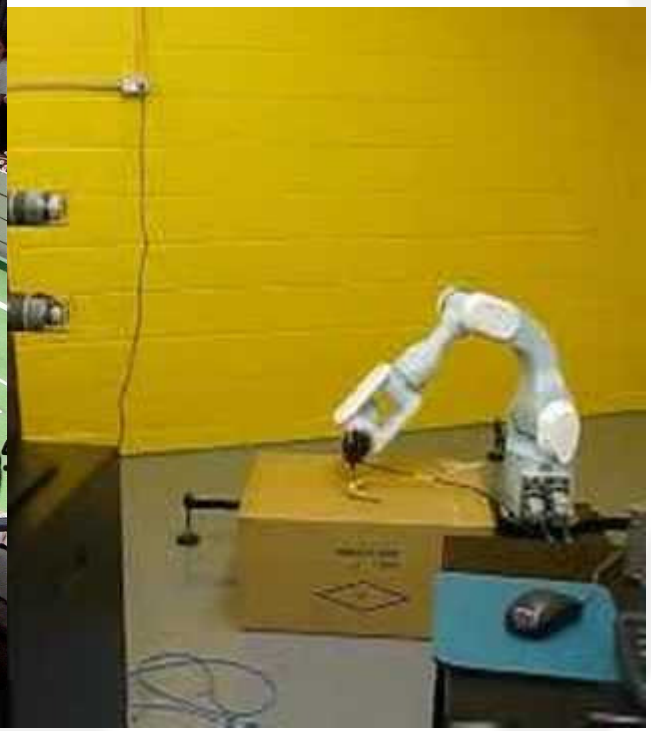
Underwater exploration

Teleoperation



Robot coordination





CONTROL PROBLEMS

The network introduces new problems in the control loop

- Transmission delay
- Packet losses
- Bit errors

Transmission delay

- Def: total time between the generation of command/measurement and its use at actuator/controller side
- Contributes to the delay:
  - Packet encoding time
  - Propagation delay
  - Channel arbitration time
  - Re-transmission time
  - Processing time by the intermediate systems (switches, access points, routers, coordinators)

Delay components

- Packet encoding time depends on packet size and speed of network interface
  - Constant for a given system/application
- Propagation delay depends on signal speed and transmission range
  - Constant for wired networks
  - Variable for wireless networks (e.g., acoustic waves)

Delay components (2)

- Channel arbitration time is the time to get the use of the channel
  - Constant for TDMA policy
  - Variable for CSMA
- Re-transmission time: the network interface waits for ack and it may retry several time before success
  - Variable and depending on the traffic level and channel quality

Delay components (3)

- Processing time by the intermediate systems
  - CPU time (constant)
  - Waiting time in queues (variable and depending on the traffic level)
  - The higher is the number of traversed intermediate systems, the higher is the delay!

Impact of delay on plant control

- The delay value determines the loop delay and should be compatible with the control application
  - Requirements on reaction time, settling time
- Delay variation alters the interval between packets (sampling frequency) and loop delay
  - Assumptions on sampling frequency and loop delay used in the control design are not satisfied at run-time

Packet losses

- Def: some packets do not arrive at destination (actuators and controller)
- Causes:
  - Channel failure (not an exception in wireless)
  - Queue overflow due to high traffic
  - Timeout of the receiver (if a sample is too much delayed then it is useless)
  - Bit errors detected (but not corrected)

Impact of packet loss on plant control

- The control loop is temporary broken:
  - Commands are not applied
  - Measurements are not considered by the controller

Bit errors

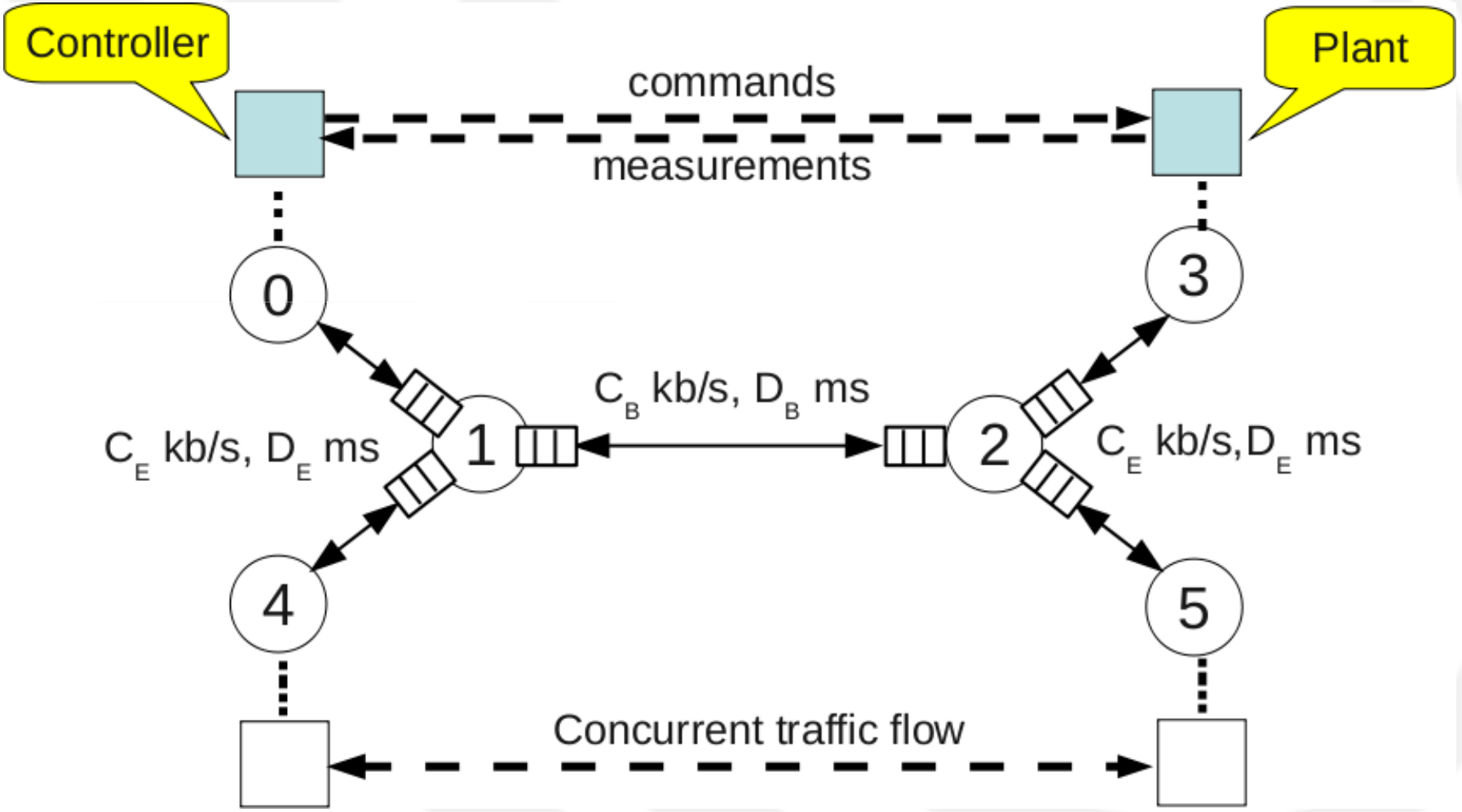
- Def: one or more bits in the packet are flipped and this event is not detected by error check
- Causes:
  - Channel failure
  - Interference from other signal sources
  - Signal weakness due to distance and obstacles

Impact of bit errors on plant control

- Command/measurement is altered
  - Let's imagine a position value with the most significant bit altered by a bit error!
- Weak SW blocks may crash in presence of un-expected values

SOLUTION 1: DIFFERENTIATED SERVICES

Network scenario



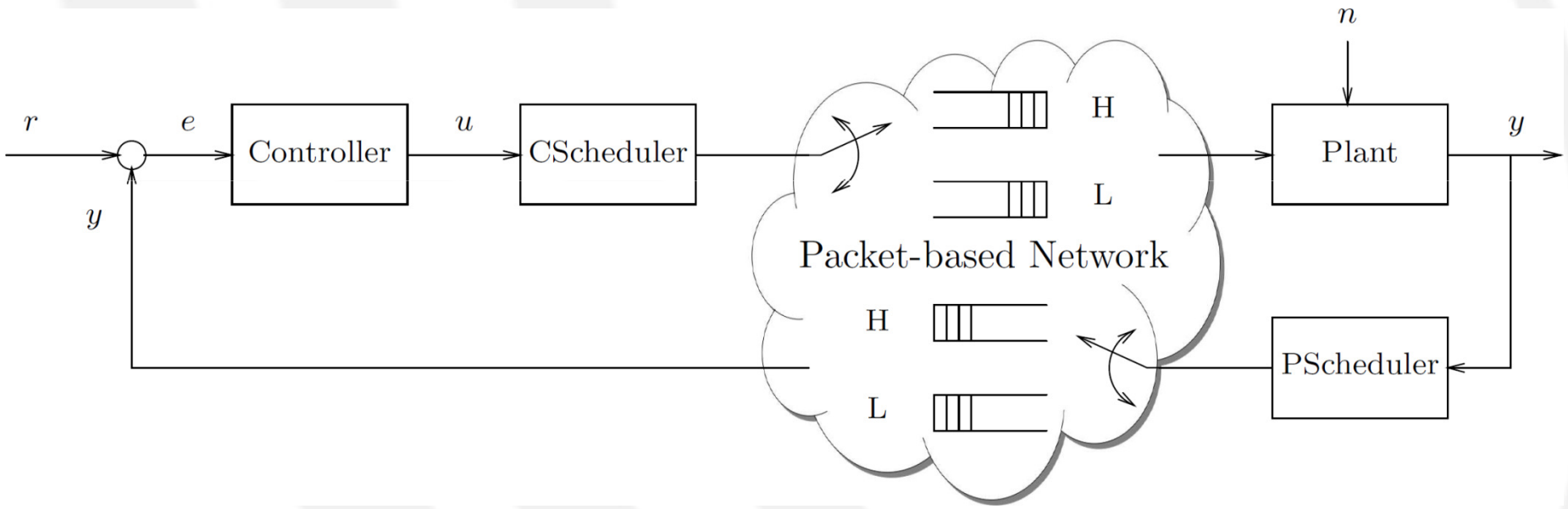
Network scenario (2)

- Example of a typical bottleneck topology in which peripheral nodes are connected through high-capacity low-delay links to a backbone link with less capacity and higher delay
- Between Node 0 and Node 3: NCS traffic (commands and measurements)
- Between Node 4 and Node 5: concurrent traffic
- Node 1 and 2 are switching nodes with input queues

Network scenario (3)

- Since the backbone capacity is shared among different traffic flows, queue level may vary inside nodes and congestions may happen leading to:
  - transmission delays
  - packet drops
- Key Observation:
  - Transmission performance depends on the design of:
    - nodes and channels
    - communication protocols
    - service differentiation
  - The perfomance of a NCS can be related not only to the controller design technique (as usual) but also to the design of the transmission strategy

Differentiated services model (DiffServ)



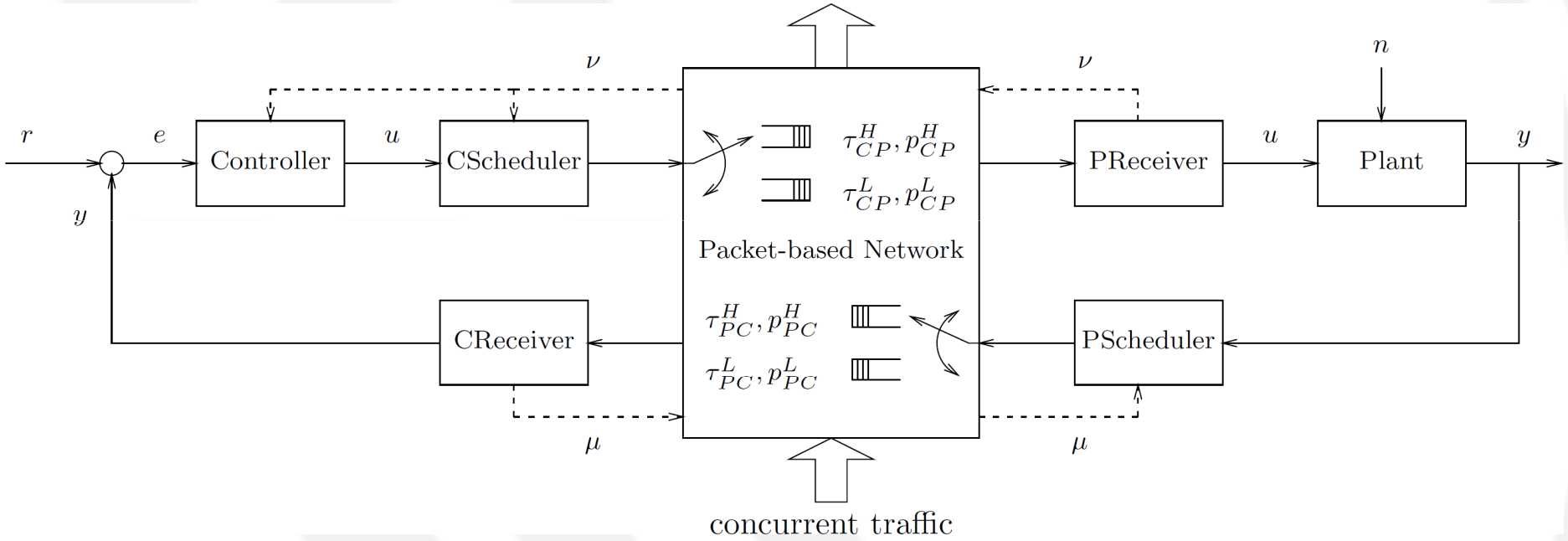
Differentiated services model (2)

- DiffServ is a standard technique to introduce Quality-of-Service (QoS) guarantees in IP networks.
- At each intermediate system there are 2 queues for 2 different forwarding priorities (virtual wires):
  - H queue for high-priority packets (H policy)
  - L queue for low-priority (best effort) packets (L policy)
- Schedulers assign priorities to packets by setting an appropriate field in their header
- Only a fraction of the total bandwidth should be devoted to the H policy otherwise differentiation becomes useless

Solution outline

- General Goal
  - Improve the performance of a Networked Control System by using the network resources in a smarter way
- Specific problem to be solved
  - Improve the control performance by optimally distributing packets between H and L policy
- Strategy for the solution
  - Mark the current packet as L or H according to the importance of the packet and the time-varying statistics of the network

Proposed architecture



Proposed architecture (2)

- Plant: continuous-time system
- Controller: discrete-time system with sample time Ts
- CScheduler (PScheduler): scheduler at the controller(plant)-side that decides the policy for the command (measurement) packets
- τ^H_{CP}, τ^L_{CP}, p^H_{CP}, p^L_{CP} and τ^H_{PC}, τ^L_{PC}, p^H_{PC}, p^L_{PC}: time-varying transmission delays (τ) and packet loss rates (p)
- CReceiver: receiver at the controller-side that computes the estimations on delay and PLR to be sent to the plant-side
- PReceiver: receiver at the plant-side that computes the estimations on delay and PLR to be sent to the controller-side

Network assumptions

- Let τ^H, τ^L and p^H, p^L be the transmission delays and packet loss rates for the H queue and the L queue, respectively.
- τ^H, τ^L are time-varying values and their estimation is computed at run time by comparing the time stamp within the packet payload and the arrival time (assuming synchronized nodes)
- p^H, p^L are time-varying values and their estimation is computed at run time by counting the arrived and lost packets.
- The following relationship holds
  - $\tau^H \leq \tau^L$
  - $p^H \leq p^L$

Network assumptions (2)

- The packets sent in each queue arrive in the proper order (this means that a packet can be overtaken only by packets belonging to a different queue)
- The delay values τ^H, τ^L and the packet loss rate values p^H, p^L are different in the controller-to-plant and plant-to-controller paths
- The delay values τ^H, τ^L are smaller than a sampling interval

Schedulers

- They choose the policy π_k for the *k*-th packet
- CScheduler: scheduler at the controller side → commands
- PScheduler: scheduler at the plant side → measurements

Algorithm for the CScheduler

- To choose the policy π_k for the *k*-th packet we go through the following steps:
  - compute the estimated plant output for the successful reception (using H and L policies) and for the lost packet case
    - we assume to hold the previous command whenever the current command does not arrive, i.e. ^u(k) = u(k-1)
  - compute the displacement between the estimated plant outputs
  - compare the displacement with a used-defined threshold to choose the policy (π_k = H or π_k = L)
- CScheduler: scheduler at the controller side → commands
- PScheduler: scheduler at the plant side → measurements

Algorithm for the CScheduler (2)

- Network condition is considered by weigthing the estimation of plant behaviour through packet loss statistics used as *a-posteriori* probabilities

$$\hat{y}^H(k) = (1 - \hat{p}^H(k))\hat{y}^H_{get}(k) + \hat{p}^H(k)\hat{y}_{lost}(k)$$
$$\hat{y}^L(k) = (1 - \hat{p}^L(k))\hat{y}^L_{get}(k) + \hat{p}^L(k)\hat{y}_{lost}(k)$$

The displacement between estimated outputs is:

$$\hat{e}(k) = \hat{y}^L(k) - \hat{y}^H(k)$$

Algorithm for the CScheduler (3)

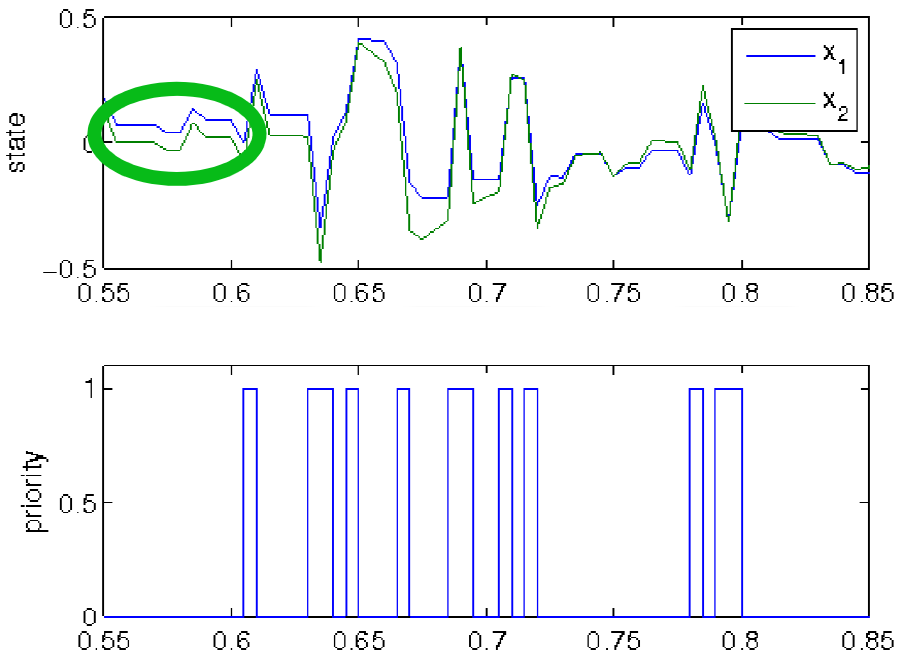
- Plant state is considered by observing measurements and estimating evolution by recording the current transmission strategy

if $|\hat{e}(k)| > E$
  then $\pi_k = H$
    $\hat{x}(k+1|\pi_k) = \hat{x}^H_{get}(k+1)$ % next state
  else $\pi_k = L$
    $\hat{x}(k+1|\pi_k) = \hat{x}^L_{get}(k+1)$ % next state

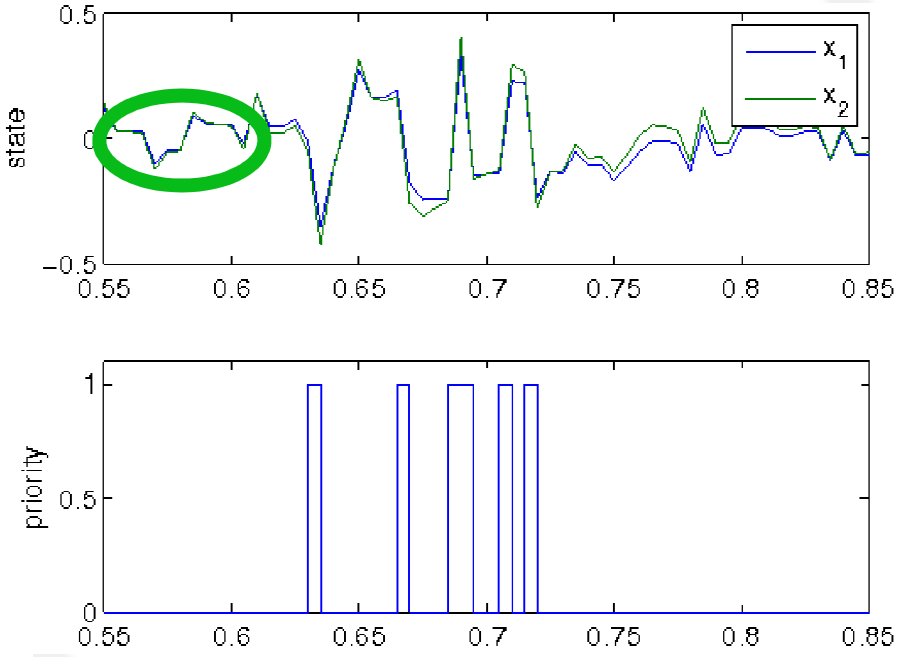
Example

- Control goal: try to keep plant state at zero
- Two different intervals in the network with different values of packet loss rate in the low-priority class

Example

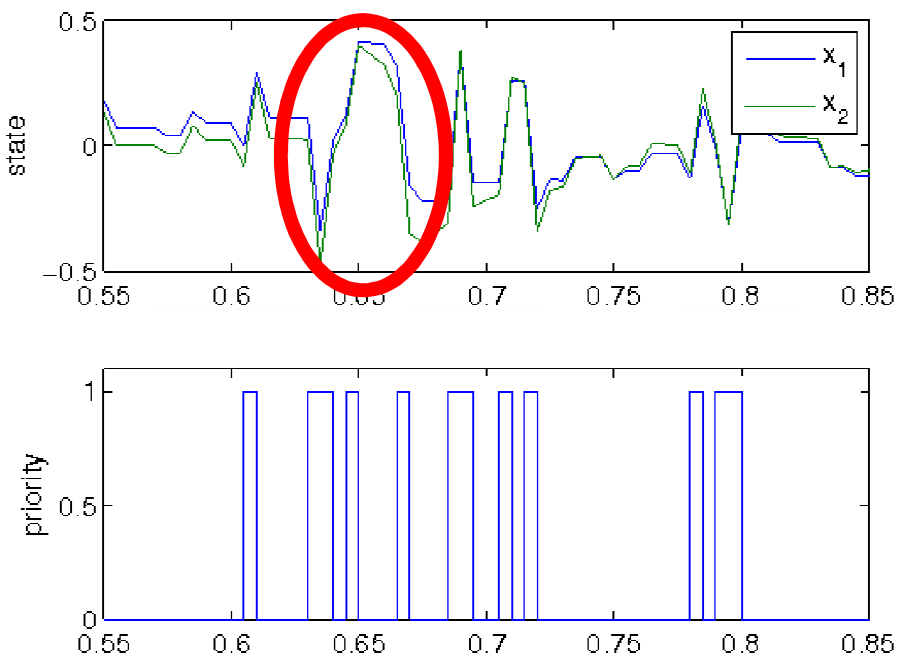


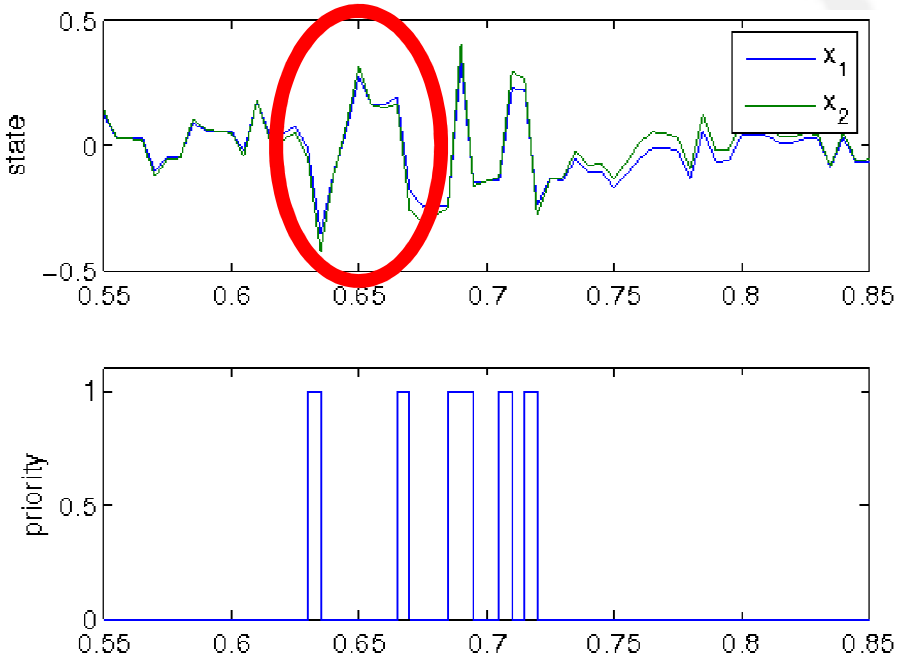
Packet loss rate
- H service: 10%
- L service: 50%



Packet loss rate
- H service: 10%
- L service: 35%

Example





Packet loss rate
- H service: 10%
- L service: 50%

Packet loss rate
- H service: 10%
- L service: 35%

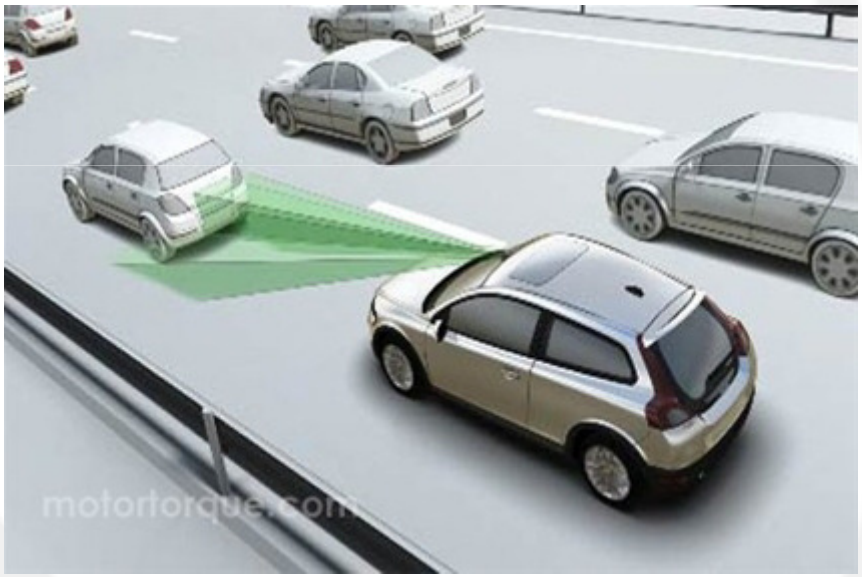
Advantages

- High priority packets are used when
  - Important data have to be sent
  - Low priority service is bad due to network problems
- High-priority class is not wasted
  - More controller-plant pairs can share the same channel since statistically they may not have to send important packets at the same time

SOLUTION 2: DESIGN OF TRANSMISSION POLICY IN A FORMATION CONTROL SCENARIO

Formation control

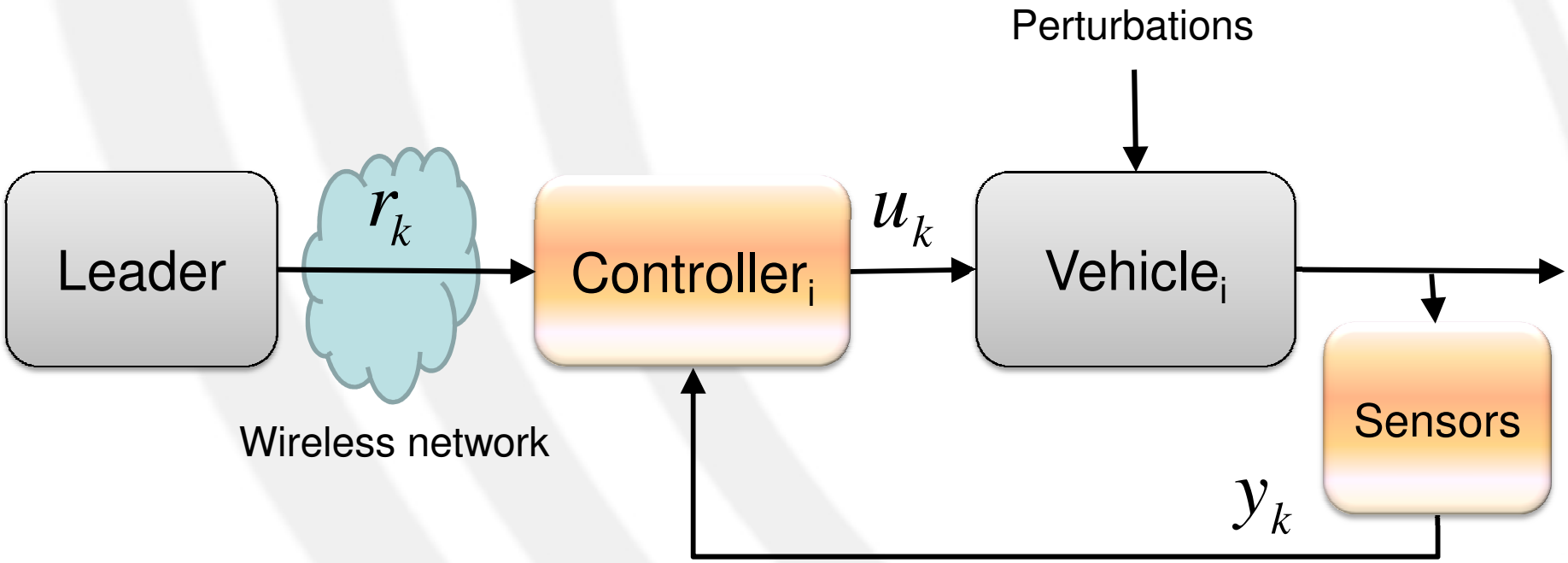
- Autonomous vehicles should adapt their trajectory and speed to keep relative distances with respect to each other
  - The formation leader chooses the trajectory
  - Presence of spatial constraints (e.g., a street) and obstacles should be taken into account



Formation control and NCS

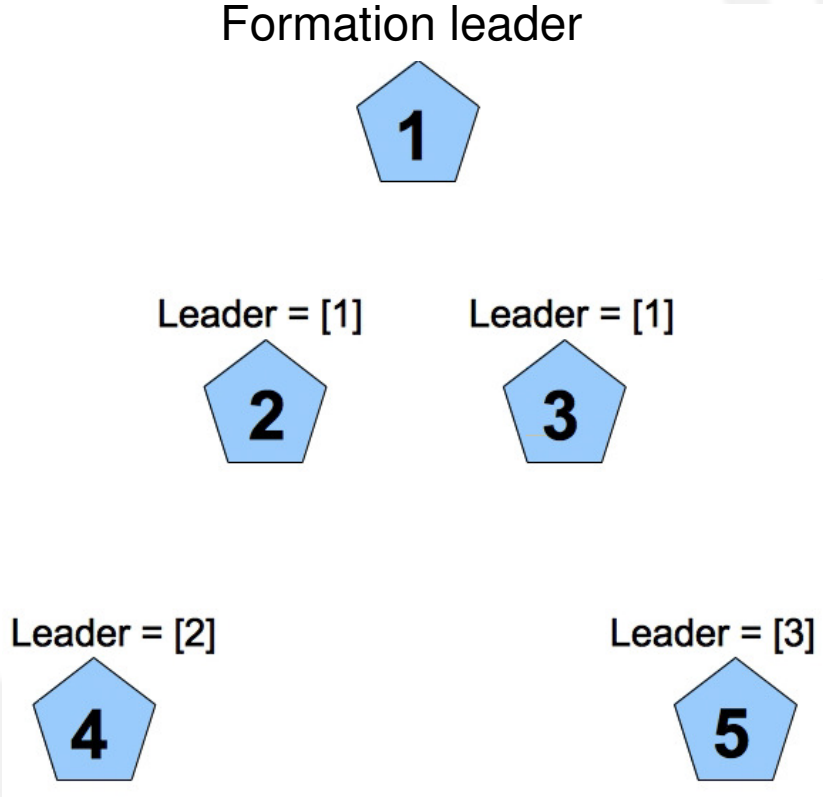
- Each vehicle receives position and speed of neighbours through wireless messages (reference signal), considers its own position and speed (measurements) and changes trajectory and speed accordingly (commands).
- Each vehicle can be considered a NCS
  - Plant: its dynamic and cinematic behavior
  - Controller and Plant are directly connected
  - Reference: the position of the leader received through the network
  - Perturbation: its position and speed may change due to wind, water flow, obstacles.

Formation control and NCS



A simplified but complete scenario

- Each vehicle
  - has only one leader
  - broadcasts its position and speed so that followers can know them
  - receives all packets but keeps only the ones from its leader
  - changes its trajectory and speed according to the speed/position of its leader



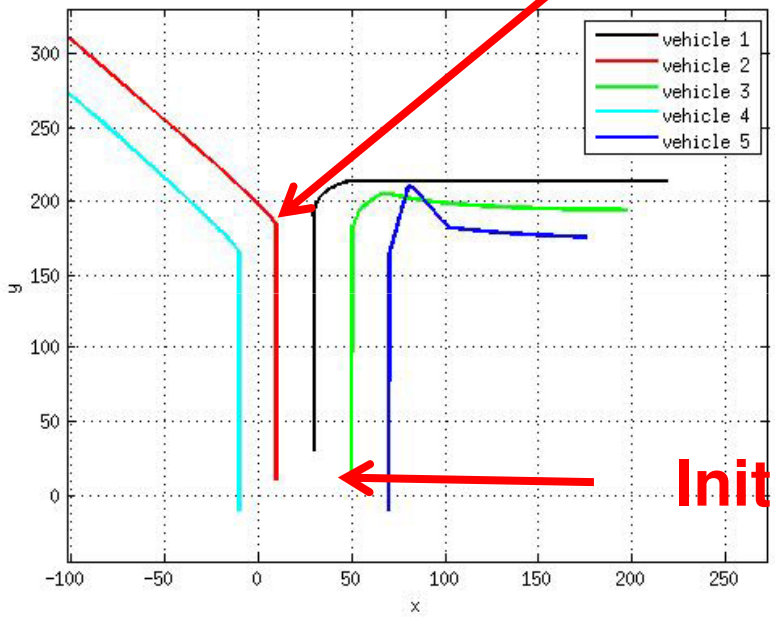
Open problems

- Messages may not arrive to the followers
  - Collisions
  - Out-of-range transmission
- If the position of the leader is not heard then the vehicle cannot react in time
  - Collisions (vehicles not packets!)
  - Loss of vehicles
- Reaction delay depends on the timely reception of reference packets
  - Channel access
  - Propagation delay as a function of distance (relevant in case of acoustic wave transmission of underwater vehicles)

Design space dimensions

- Controller: it takes the position/speed errors and decides the speed
  - Traditional control design approach
- Network: many parameters can be controlled
  - Transmission power
  - Channel access policy (TDMA vs. CSMA)
  - Packet priority
  - Packet redundancy

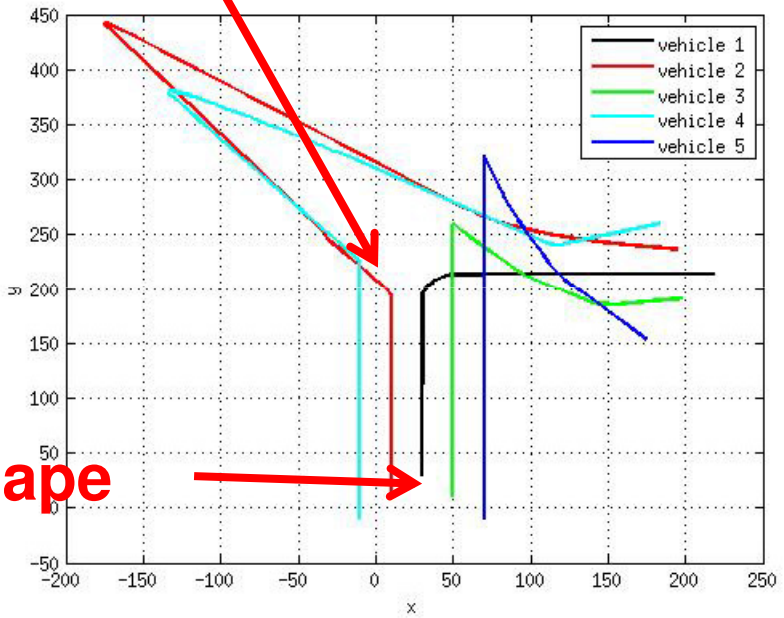
Transmission power



Always minimum power

Perturbation

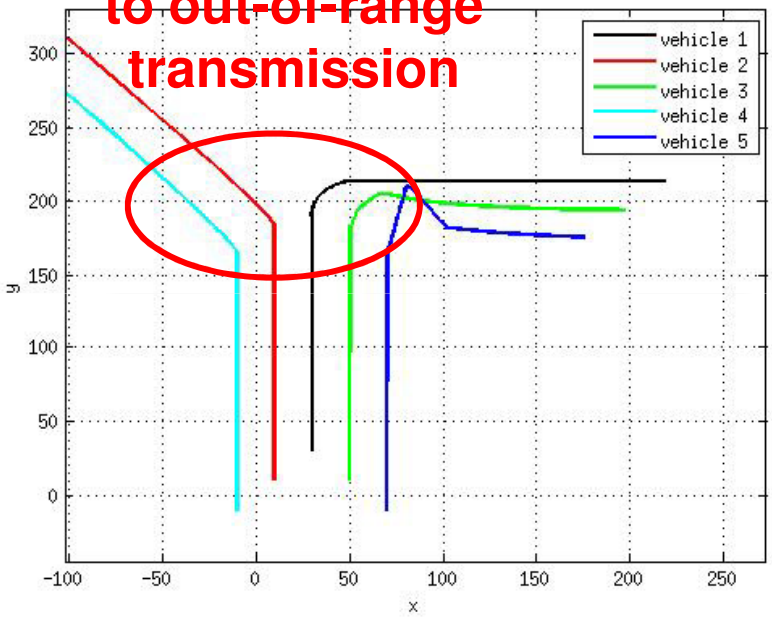
Initial shape



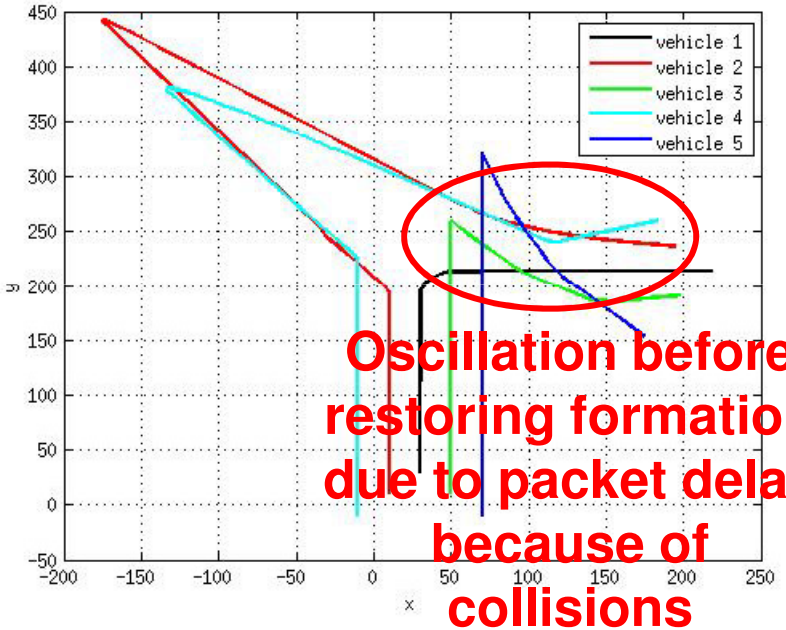
Always maximum power

Transmission power

Loss of vehicles due to out-of-range transmission



Always minimum power



Oscillation before restoring formation due to packet delay because of collisions

Always maximum power

Power adaptation is desired !!!

Transmission power adaptation

- Goal: adjust the transmission power to reach the followers (no more!) avoiding to disturb others
- Strategy: use the information received by followers to estimate their distance and adjust TX power accordingly
  - not used for speed control
  - used to change the transmission power according to a given *signal loss law*

Signal loss law

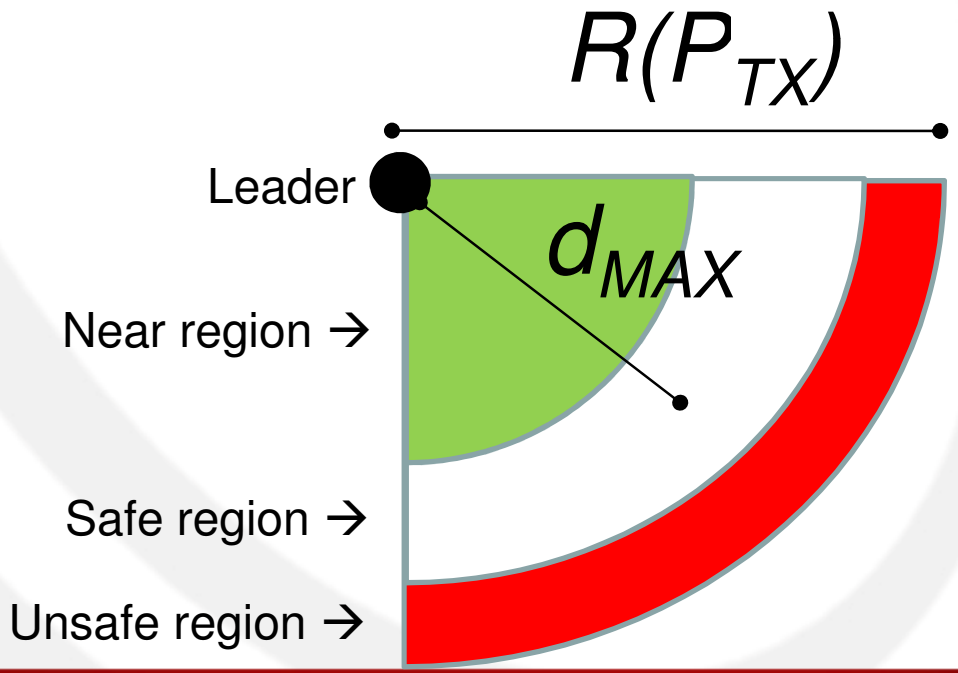
- Let us assume that the signal power decreases according to

$$P_{RX} = \frac{1}{d^{\alpha}} P_{TX}$$

- If $P_{RX} < P_{min}$ then the packet is not received
- Let $R(P_{TX})$ the valid transmission range in which $P_{RX} \geq P_{min}$

Power adaptation algorithm

- At each sample time:
  - hear position messages from the followers
  - let d_{MAX} be the maximum distance among all the followers
  - adjust P_{TX} so that d_{MAX} falls in the *Safe Region*



SIMULATION ISSUES FOR NCS

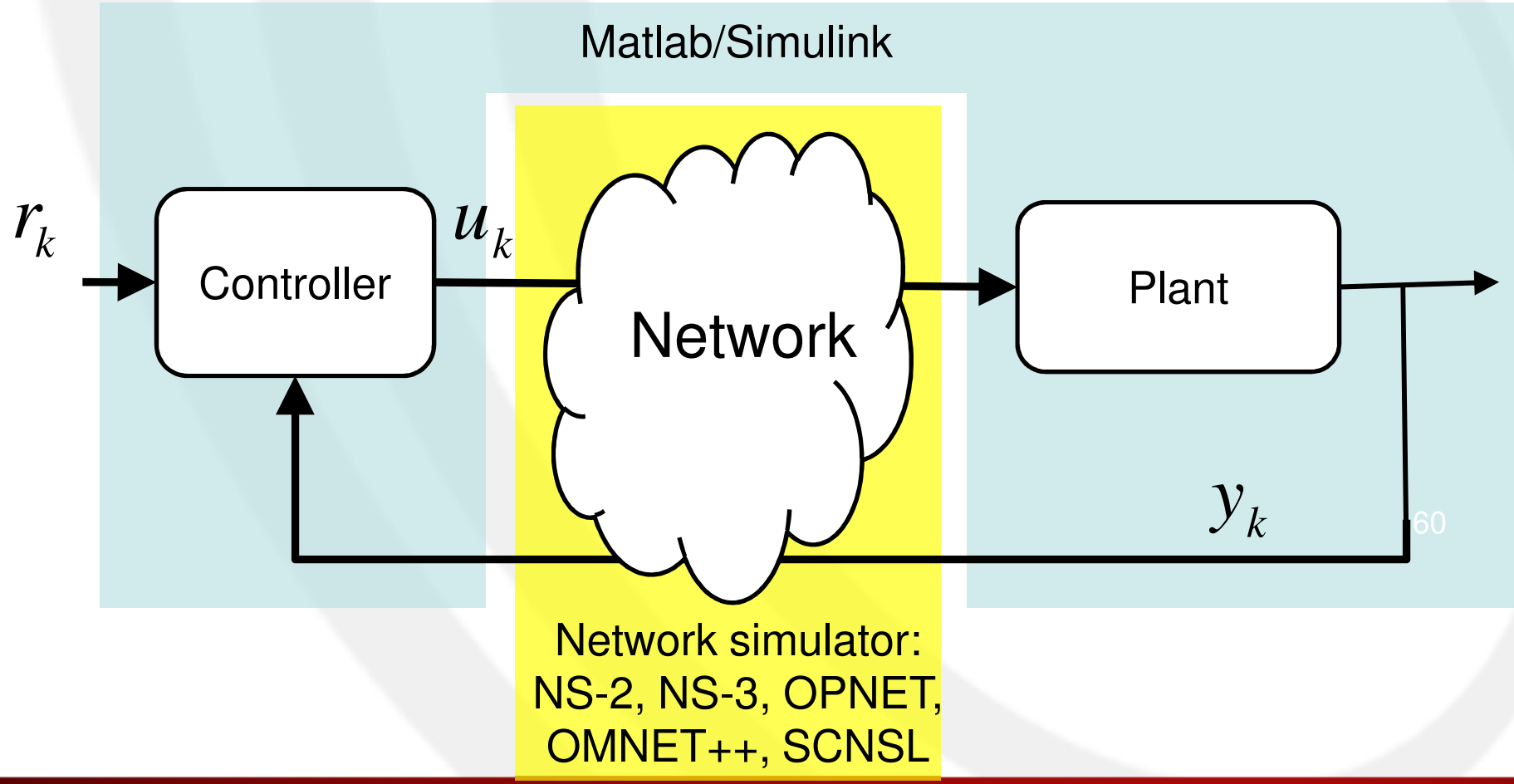
Simulation of controller and plant

- Matlab/Simulink ← very popular
- Labview
- Modelica
- 20-sim

Simulation of the network

- Matlab/Simulink
  - Very accurate models at physical level
  - At packet-level just injection of delay and packet losses on a point-to-point link
- Network simulator (NS-2/3, OPNET, OMNET++, SCNSL)
  - Description of a full topology
  - Effect of interaction with other packet sources
    - Collisions
    - Hidden node phenomenon

Co-simulation



Co-simulation issues

- How to connect different tools?
  - Socket
  - Shared memory
- How to synchronize simulation threads?
  - Only one tool execution at each time
  - Parallel execution with synch points
- How to represent external entities inside the model?

Single hybrid model

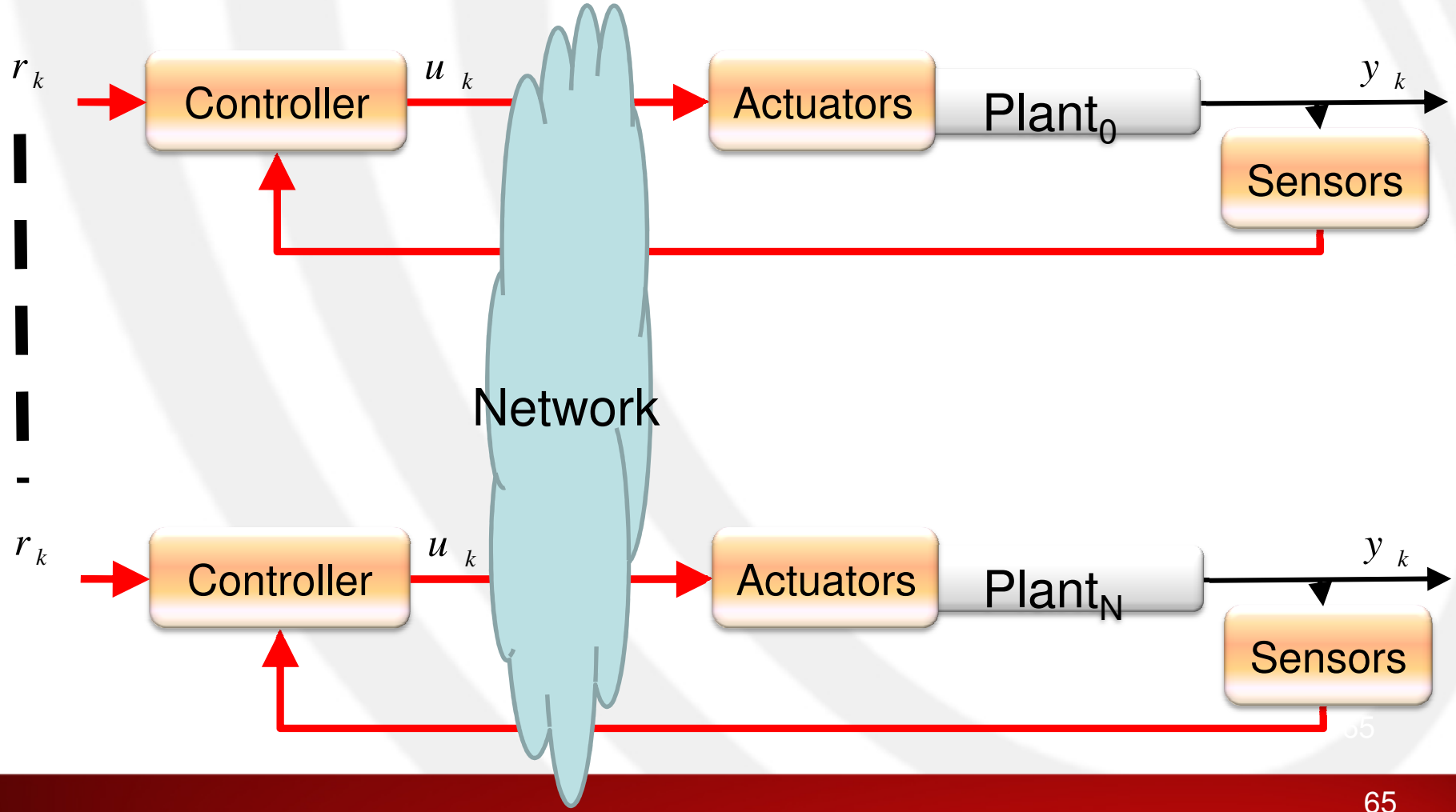
- Description of the NCS as a hybrid automaton (model of computation)
  - States capture continuous evolution
  - Transitions capture discrete evolution
- Simulation by generating a single execution model
  - SystemC, C++, ANSI C
  - Faster than co-simulation
  - Possible parallelization on a multi-core host (e.g., GP-GPU, CUDA, etc.)

Single hybrid model: issues

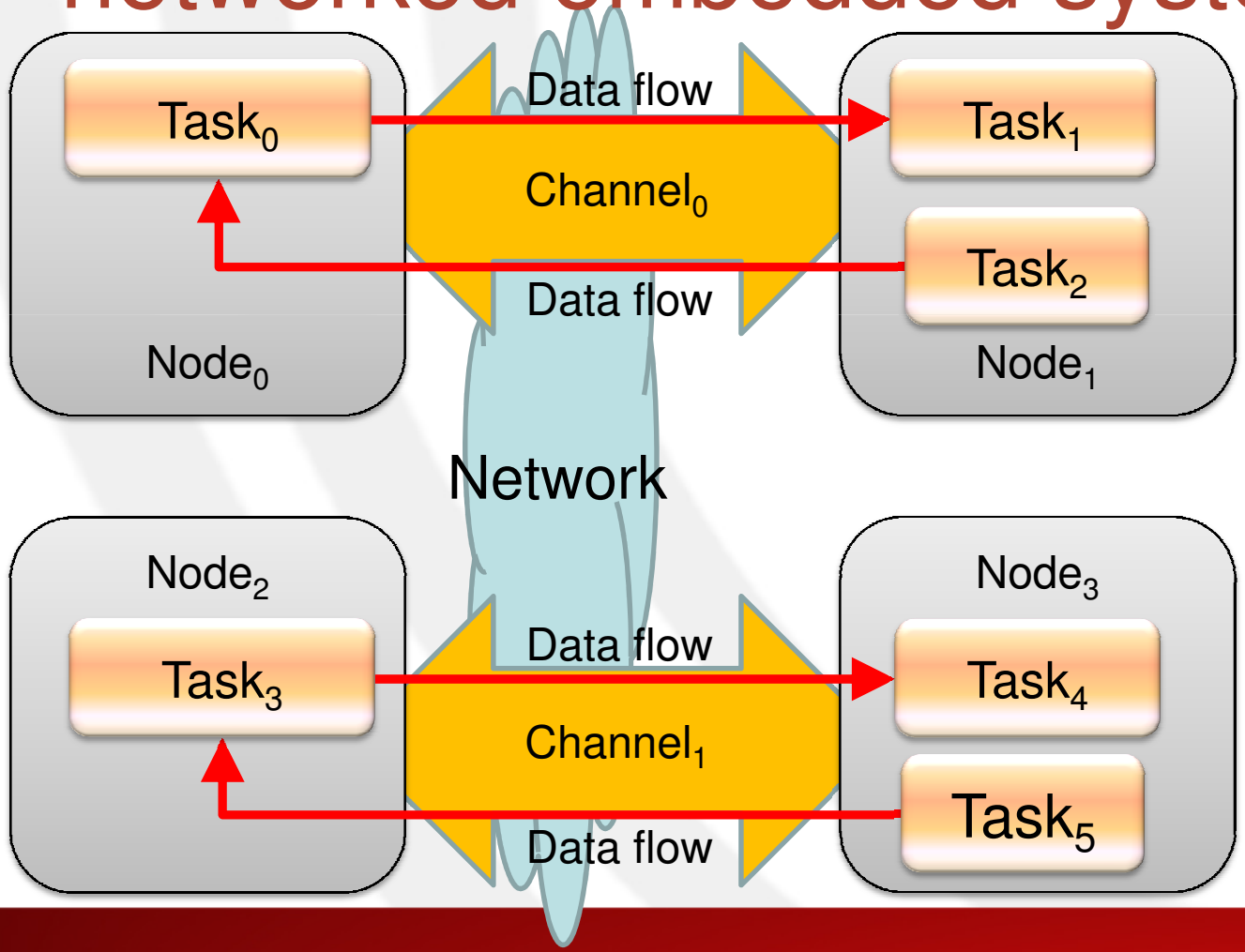
- Designing a hybrid automaton from scratch is not user-friendly
- A tool is needed to convert models from different domains into the common representation
- The conversion tool or the simulation kernel should address optimal discretization of continuous aspects

RELATIONSHIP BETWEEN NCS DESIGN AND NES DESIGN

Set of networked control systems



Distributed application of networked embedded systems



NCS-NES relationship

- The NCS is a particular application of NES
- Controller, actuators and sensors are tasks hosted on networked embedded systems
- Commands and measurements are data flows
- The network consists of one (shared) or more (dedicated) channels

From NCS design to NES design

- Step 1) design of control strategy and communication strategy
  - Joint design is preferable
- Step 2) mapping of controller, actuators, sensors and network on actual embedded systems
  - HW and SW for computation, actuation, sensing and communication