Motion Control and Real-Time Systems: an Approach to Trajectory Rebuilding in Non-Deterministic Networks

Manuel Spera
manuel.spera@unibo.it

Bologna, June 20, 2008
DEFINITION

*Real-time System:* System in which the correct behavior of the whole depends not only on the logic results of the operations, but also on the instant time in which these results are produced.

DEFINITION

*Motion Control:* Sub-field of automation, in which the position and/or velocity of machines are controlled using some type of device such as a hydraulic pump, linear actuator, or an electric motor, generally a servo. In motion control the position, velocity, force, pressure, etc., profiles are designed in such a way that the different mechanical parts work as an harmonious whole in which a perfect synchronization must be achieved.
In industrial automation:

Field Area Network (Fieldbus) → They link sensors, actuators and controllers (Ex. PLC), man-machine interface.

Main features of the transferred data:

- low data rates
- small data packets
- real-time capabilities required

Deterministic communication system

But other kind of data (configuration parameter, file transfer) and traffic (best effort) can pass through a net...

Many protocols, many standards.

(IEC61158, IEC61784, IEC62026, EN50170, EN50254, EN50325 covering CAN, Profibus, Profinet, WorldFip, P-Net, Lonworks, ControlNet, CANOpen...)
The most important feature that a real-time communication system must have is the preservation of the following properties:

- **Absolute Temporal Consistency:** It refers to the difference in time between the current time and the time at which the information has been acquired.

- **Relative Temporal Consistency:** It applies when samples from different signals must be correlated in time.

- **Spatial Consistency:** It applies when the same information is copied at different locations.
This work deals with the preservation of the temporal properties of a profile transmitted adopting a non-deterministic network.

The following cases will be faced:

1. Master-slave
2. Multi-slave
3. Multi-master (or Gateway)

These are the key elements used for the construction and analysis of more complex networks.
Case 1: Master-Slave

ET SYSTEM

- Trajectory $x(t)$
- Hypothesis: the first data generation time corresponds to the time origin
- Sampling:
  $$x_{M,k} = x((k-1)T_M) \quad (k = 1, \ldots, \infty)$$

- Transmission delay:
  $$\Delta_t + \text{jitter}(k)$$

- Received samples:
  $$x_{S,k}(t_{a,k}) = x_{M,k}$$
  where:
  $$t_{a,k} = (k-1)T_M + \Delta_t + \text{jitter}(k)$$

- Samples' generation:
  $$x_r((r-1)T_S) = f(x_{S,k}, x_{S,k-1}, \ldots)$$
Case 1: Master-Slave

In particular:

The data absolute temporal consistency is lost.
Two sub-problems:
1. Master clock recovery (local)
2. Trajectory rebuilding (based on the information provided by the previous step).
Clock master

Clock generated by the data arrival on the slave node

Clock generated by the ideal data arrival on the slave node

Desired clock generated by the data arrival on the slave node
Ideal Clock

Desired Clock

Ideal desired clock rebuilt by means of the counter

Real clock rebuilt adopting the proposed control strategy
A phase-locking technique is adopted.

Useful in case of:
- jitter reduction
- clock alignment
- frequency synthesis
- clock recovery.

Usually implemented in the lower layer (physical layer) of the communication system to recover damaged information.

In this talk, a PLL is implemented on a higher level of abstraction.
Master Clock Recovery

PHASE-LOCKED LOOP

Basic idea:
- a counter is used to regenerate the master clock
- the reference of the control loop is the measured phase of the incoming signal

The jitter is handled as high frequency noise

The controller is designed as a low-pass filter

CLASSIC SCHEME

\[ x(t) = A \sin(\omega_c t + \varphi_n(t)) \]

\[ y(t) = A \sin(\omega \int \phi \, dt + K_{\text{VCO}} \int_{-\infty}^{t} V_{\text{cont}} \, dt) \]

It allows:
- frequency acquisition (locked state, \( \delta \varphi = \text{const} \))
- phase acquisition
Master Clock Recovery

\[ n_s = n_{\text{reset}} + n_{\text{jitter}}(k) - \alpha n_{\text{reset}} \]

• Reference:

\[ \delta \phi = \bar{n}_{\text{count}} \]

• Plant:

\[ n_{\text{count}}(z) = \frac{1}{1-z^{-1}} (n_{\text{reset}} - n_{\text{reset}}(z)) \]

• Controller:

\[
\begin{align*}
\bar{n}_{\text{reset}}(k) &= a \bar{n}_{\text{reset}}(k-1) + (1-a) \left[ n_{\text{reset}}(k-1) - n_{\text{count}}(k-1) + n_{\text{count}}(k) \right] \\
n_{\text{reset}}(k) &= \bar{n}_{\text{reset}}(k) - \text{gain} \bar{n}_{\text{count}} \\
\end{align*}
\]

\[
\frac{N_{\text{reset}}(z)}{N_{\text{count}}(z)} = (1-a+\text{gain}) \cdot \frac{1-(1-a+\text{gain})z^{-1}}{1-z^{-1}}, \text{ that is a PI}
\]
Trajectory Rebuilding

All the parameters useful for the trajectory rebuilding are available.

ALGORITHM

1. The Slave node waits for two data available in the buffer

2. At the first slave data request, and at every subsequent request, the actual value of the counter $n_{\text{count}}$ is read

3.a If a counter’s reset is not already happened, a linear interpolation is performed adopting as extreme points the first two data present in the buffer on the basis of the actual values $n_{\text{count}}$ and $\overline{n}_{\text{reset}}$

3.b If a counter’s reset is already happened, the buffer is updated adopting a FIFO policy and a linear interpolation is performed adopting as extreme points the new first two data present in the buffer on the basis of the actual values $n_{\text{count}}$ and $\overline{n}_{\text{reset}}$

INTERPOLATION FORMULA

$$x_r = \begin{cases} \frac{n_{c,r}}{n_{\text{reset}}(k-1)}(\hat{x}_{S,k} \cdot \hat{x}_{S,k-1}) + \hat{x}_{S,k-1} & \text{if } n_{c,r} < n_{\text{new},k} \\ \frac{n_{c,r}}{n_{\text{reset}}(k)}(\hat{x}_{S,k} \cdot \hat{x}_{S,k-1}) + \hat{x}_{S,k-1} & \text{if } n_{c,r} \geq n_{\text{new},k} \end{cases}$$
Master-Slave: Simulation Results

DATA

- Trajectory: $x(t) = 2000 \sin(8\pi t)$
- nominal $T_M = 2$ msec;
- nominal $T_s = 1$ msec
- $T_{clk} = 400$ nsec;
- $\Delta_r = \frac{1}{4}$ nominal $T_M$;
- Jitter: stochastic variable having uniform distribution with maximum value equal to 0.2 msec;
- Initial $n_{\text{reset}} = \frac{\text{nominal } T_M}{T_{clk}} = 5000$;
- Initial $n_{\text{count}} = \frac{3}{4}$ of initial $n_{\text{reset}} = 3750$;
- Simulation length = 0.5 sec;

- Controller's parameter: $a = 0.96907$, $\text{gain} = 0.032334$.

A priori, the jitter is unknown... How can we decide where the bandwidth has to be cut?

COMPROMISE
We cut the bandwidth as low as possible compatibly with the acquisition velocity and hardware implementation
Master-Slave: Simulation Results

Real $T_M=95\%$ nominal $T_M$, real $T_S=nominal$ $T_S$

Rebuilding based on the nominal values

Rebuilding by means of the algorithm

- The rebuilding error increases
- The rebuilding error decreases
Master-Slave: Simulation Results

Rebuilding by means of the algorithm

- Interrupt generated by the incoming data
- Counter's reset vs. Ideal reset
- Time error
Master-Slave: Simulation Results

Rebuilding by means of the algorithm

Value of the counter's reset computed when interrupt arrives

Average value of the counter's reset computed when interrupt arrives
Master-Slave: Simulation Results

- The buffer accumulates data

- The buffer is under control

Rebuilding based on the nominal values

Rebuilding by means of the algorithm

- The buffer is under control

Flow control!!
Case 2: Multi-Slave

- The Master node has to transmit different trajectories to different Slave nodes
- A temporal relation exists among the trajectories

The relative temporal consistency property must be guaranteed
Hypothesis: a broadcast signal, generated by a master node, is received by all the slave nodes at the same instant time.

**Procedure on the master node:**

- Broadcast Message
- Msg 1 to Slave 1
- Msg 1 to Slave 2
- Msg 1 to Slave n
- Broadcast Message
- Msg 2 to Slave 1
- Msg 2 to Slave 2

**Basic Cycle**

\( T_m \)

**Procedure on the slave node:**

- The slave waits for the broadcast signal
- The slave waits for the sampled data of the trajectories
- The reconstruction is based on the arrival time of the broadcast signal and not on the data arrival time (always adopting the previous PLL-based algorithm)
Multi-Slave: Simulation Results

DATA

- Trajectory A: \( x_A(t) = t \)
- Trajectory B: \( x_B = \sin(8\pi t) \)
- Trajectory C: \( x_C = \cos(8\pi t) \)
- Nominal \( T_M = 1 \) msec;
- Nominal \( T_{s,A} = 3.5 \) msec;
- Nominal \( T_{s,B} = 2.4 \) msec;
- Nominal \( T_{s,C} = 2.2 \) msec;
- \( T_{\text{clk},ABC} = 1 \) \( \mu \)sec;
- \( \Delta_r = \frac{1}{4} \) nominal \( T_M \);
- **Jitter**: stochastic variable having uniform distribution with maximum value equal to 0.1 msec;
- Simulation length = 0.8 sec.

Remark: in this case, the phase displacement \( \alpha T_M \) is not required.
Multi-Slave: Simulation Results

3D rebuilt trajectory:

Original (red) and rebuilt (blue) trajectories

RMS Rebuilding error

Trajectory A

Trajectory B

Trajectory C

Time

RMS error

0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

0
10^-7
1
10^-6
10^-5
10^-4
10^-3
10^-2
10^-1
1
Case 3: Multi-Master

- One slave (gateway) receives data (trajectories) from two, or more, masters
- Again, a temporal relation exists among the trajectories

Aim of the gateway is the restoring of the correct temporal relation among the trajectories
Hypothesis:
1. The gateway is able to identify the first received data describing a trajectory transmitted by a master node;
2. The first data of every trajectory must be placed in the same time instant.

Procedure:
• The first two data of a trajectory are necessary to start the trajectory rebuilding, so the gateway has to wait for two data for every trajectory before the begin of the trajectory rebuilding algorithm.
• Anyway, the clock alignment to one master immediately starts adopting the PLL-based algorithm.
• When the “last” second data arrives, the regenerated master clocks are subject to a displacement in such a way to align all the second data receiving time.
• The trajectory rebuilding starts.
Multi-Master: Proposed Solution

**Ideal case:**
- Master A
  - A,1
  - A,2
  - A,1_Data
- Master B
  - B,1
  - B,2
  - B,3
  - B,1_Data
  - B,2_Data
- Gateway G
  - A,1
  - A,2
  - A,1_Data
  - A,2_Data

**Real case:**
- Master A
  - A,2
  - A,1
  - A,1_Data
  - \( t_{A,1S} \)
  - \( t_{A,2S} \)
  - \( t_{A,2} \)
  - \( t_{A,2S} \)
- Master B
  - B,1
  - B,2
  - B,3
  - B,2
  - B,2_Data
  - B,1_Data
  - \( t_{B,1S} \)
  - \( t_{B,2S} \)
  - \( t_{B,3S} \)
- Gateway G
  - A,1
  - A,2
  - Align,1
  - Align,2
  - \( t_{align,1S} \)
  - \( t_{align,2S} \)

\( \alpha T_m \)

Ideal case:
- Master A
  - A,1
  - A,2
  - A,1_Data

Real case:
- Master A
  - A,2
  - A,1
  - A,1_Data
  - \( t_{A,1S} \)
  - \( t_{A,2S} \)
  - \( t_{A,2} \)

Jitter:
\( \text{Jitter}_A(2) + \text{Jitter}_B(2) \)
Multi-Master: Simulation Results

DATA

• Trajectory $M_1$: $x_A(t)=t$
• Trajectory $M_2$: $x_B=\sin(8\pi t)$
• Nominal $T_{M,1}=2$ msec;
• Nominal $T_{M,2}=1.5$ msec;
• Nominal $T_G=1.8$ msec;
• $T_{clk}=1$ µsec;
• $\Delta_r(M_1)=\frac{1}{4}$ nominal $T_{M,1}$;
• Offset $(M_1)=0$;
• Offset$(M_2)=3.625$ msec;
• $\Delta_r(M_2)=\frac{1}{4}$ nominal $T_{M,2}$;
• Jitter $(M_1)$: stochastic variable having uniform distribution with maximum value equal to 0.18 msec;
• Jitter $(M_2)$: stochastic variable having uniform distribution with maximum value equal to 0.1 msec;
• Simulation length = 0.3 sec.
Multi-Master: Simulation Results

Alignment:

Offset error! It’s a jitter effect…
Multi-Master: Simulation Results

Rebuilt trajectory:

Original (red) and rebuilt (blue) trajectories: A

Rebuilding error: A

Offset error!
Multi-Master: Simulation Results

Rebuilt trajectory:

Original (red) and rebuilt (blue) trajectories: B

Rebuilding error: B
Multi-Master: Simulation Results

Flow control:
Conclusions

• The key elements for the construction of more complex event-triggered distributed system are presented, paying particular attention to the temporal properties of the transmitted data

Open issues:

• How can we reduce the offset error in the multi-master case?
• If the net is a multi-level net, how will it work?
• Interfacing with other standards…
REAL-TIME SYSTEMS


• J.D.Decotignie, “Ethernet-Based Real-Time and Industrial Communications”, Proceedings of the IEEE, Vol.93, NO.6, June 2005

PLLs


