Intelligent distributed systems

Daniele Fontanelli

Department of Industrial Engineering
University of Trento
E-mail address: *daniele.fontanelli@unitn.it*

2022/2023

Outline

Outline

What is a networked control system?

- What we have seen so far are *Distributed Control Systems* in which, essentially, local controllers are networked to ensure a correct course of actions.
- Usually, a *centralised supervisory controller* modifies the executions of the local controllers modifying their behaviours, e.g., changing their reference or tuning parameters, or aggregating the measures, e.g., Kalman filter.
- Nevertheless, there is the possibility to avoid the presence of a *master* ruling the executions and, hence, having a fair distribution of the decisions among the local controllers.
- This is the paradigm of the *networked control systems*.

Networked control systems

Examples of networked control systems are:

- Wireless sensor networks;
- Swarm robotics;
- Communication networks;
- Next generation smart grids;
- Water distribution;
- Ground or air traffic control.

Networked control systems

Examples

Examples taken from various scientific contexts:

- *Statistical mechanics*: The local interactions of millions of particles may yield simple thermodynamics laws describing the global behaviour.
- *Cooperation*: Simple global behaviours are obtained from local interactions. One example is *flocking*: collective animal behaviour is given by the motion of a large number of coordinated individuals.
- *Social networks*: Individual social interactions produce global social phenomena.
- *Economic networks*: Economic entities take part to the global market producing global behaviours, e.g., world wide economic crisis.

Networked control systems

- The objective is the study of the behaviour of complex systems constituted by the *interconnection of many units* which are themselves dynamical systems.
- The behaviour of these systems will depend on the *dynamics of the units* and on the *interconnection topology*.
- In general, the main purpose is to study how the topology and the dynamic systems produce the *global dynamics*.
- In this course, we will limit the analysis to *distributed estimation* and to *distributed control* for very specific (and simple) systems.

Networked control systems

Graphic representation

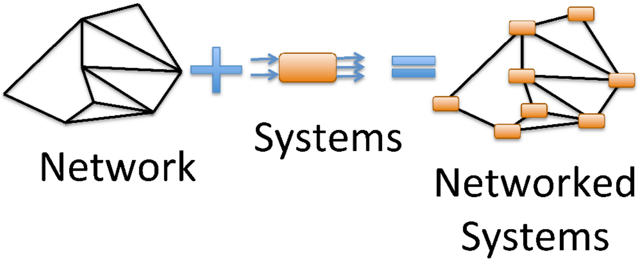


Figure: The graph representation of the networked control system: a network whose nodes are dynamic systems.

Networked control systems

In general, two main problems can be tackled for this class of systems:

- *Distributed estimation*: Using the "opinion", e.g., local measurement, of each system, construct a global estimate of the quantity of interest through *messages exchange*.
- *Distributed control*: Using the "local understanding" upon the relative configuration of each system, e.g., local measurement, solve the problem of coordinated motions, e.g., rendezvous, deployment, etc., through *messages exchange*.

Networked control systems

Linear Consensus

- One of the most promising tools are the *linear consensus algorithms*, which are simple distributed algorithms to compute averages of local quantities.
- These algorithms stem form the analysis of *Markov chains* and have been applied in the 80s to the computer science community for *load balancing*.
- In the recent years, they have been adapted and applied to cooperative coordination of multi-agent systems.
- Alternative naming conventions:
  - Agreement problems (social networks, economy);
  - Synchronisation (statistical mechanics);
  - Society of robots, rendezvous, coordinated motion (robotics).

Outline

Distributed estimation

Linear Consensus

Let us consider two sensors measuring the constant quantity x affected by the same zero mean noise, i.e.

$$z_1 = x + \varepsilon \text{ and } z_2 = x + \varepsilon$$

The Least Squares solution would be the arithmetic mean

$$\hat{x}^{LS} = \frac{\sum_{i=1}^{n} z_i}{n} = x + \frac{\sum_{i=1}^{n} \varepsilon}{n}.$$

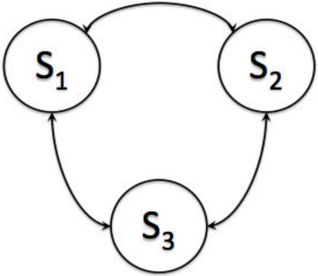
Distributed estimation

Linear Consensus

- Let us now consider the problem to compute the arithmetic mean in a *distributed way*.
- For example, consider a network comprising $n = 3$ sensors measuring the temperature of a given environment.
- The communication is bidirectional between the nodes and each node has visibility of all the other nodes of the network.

Distributed systems

Linear Consensus



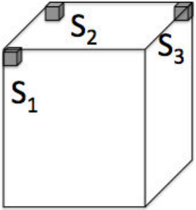


Figure: Sensors communication links and sensors deployment.

Distributed estimation

Linear Consensus

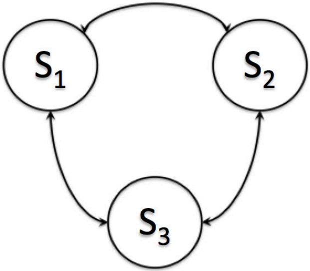
- Once S_1 receives the temperature message from S_2 and S_3 (with a proper *timestamp*), it can compute the mean.
- Similarly, S_2 and S_3.
- A convenient way to represent this distributed operation is to collect the value measured by each sensor, say $x_i(k)$, in a column vector, i.e. $x(k) = [x_1(k), x_2(k), x_3(k)]^T$, and then write the temperature update with

$$x(k+1) = Qx(k) = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} = \begin{bmatrix} \frac{\sum_{j=1}^{3} x_j(k)}{3} \\ \frac{\sum_{j=1}^{3} x_j(k)}{3} \\ \frac{\sum_{j=1}^{3} x_j(k)}{3} \end{bmatrix}.$$

where Q is called the *transition matrix*.

Distributed systems

Linear Consensus



$$Q = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}$$

Figure: Sensors communication links and algebraic representation.

Distributed estimation

Linear Consensus

- It is easy to see that being $x(0)$ the value of the temperature at the beginning, after one *protocol* iteration, i.e. after three broadcasts, one for each node S_i, the *agreement* is reached, i.e. *each node has the same mean value stored*.
- Moreover, the idea can be extended to an arbitrary number of nodes n.
- Notice that this algorithm is *distributed* since each node reaches the mean by simply knowing its own value and the value of its *neighbours*, i.e. *nodes that can share the information with node S_i*.

Distributed estimation

Linear Consensus

- To highlight the distributed nature of the protocol, the update equation can be rewritten as

$$
\begin{aligned}
x_i(k+1) &= \frac{\sum_{j=1}^n x_j(k)}{n} = \frac{1}{n} x_i(k) + \sum_{j=1, j\neq i}^{n} \frac{1}{n} x_j(k) = \\
&= \left(1 - \frac{n-1}{n}\right) x_i(k) + \sum_{j=1, j\neq i}^{n} \frac{1}{n} x_j(k) = \\
&= x_i(k) + \sum_{j=1}^{n} \frac{1}{n} (x_j(k) - x_i(k)) = \\
&= x_i(k) + \sum_{j=1}^{n} q_{ij} (x_j(k) - x_i(k))
\end{aligned}
$$

Distributed estimation

Linear Consensus

- Let us make a closer look to this distributed estimation protocol. Let $x(0)$ be the value of the temperature measured by three sensors.
- After one iteration of the protocol, one has:

$$x(1) = Qx(0) = \begin{bmatrix} \frac{\sum_{j=1}^{3} x_j(0)}{3} \\ \frac{\sum_{j=1}^{3} x_j(0)}{3} \\ \frac{\sum_{j=1}^{3} x_j(0)}{3} \end{bmatrix} = \frac{\sum_{j=1}^{3} x_j(0)}{3}\mathbf{1} = \beta\mathbf{1},$$

  where $\mathbf{1}$ is a vector of ones.
- What happens when another round of messages is broadcasted? For $x(2)$ we have:

$$x(2) = Qx(1) = \beta Q\mathbf{1} = \beta\mathbf{1}.$$

Distributed estimation

Linear Consensus

- It then follows that $\beta\mathbf{1}$ remains constant, no matter what is the number of messages exchanged.
- In fact:

$$
\begin{aligned}
x(1) &= Qx(0) = \beta\mathbf{1},\\
x(2) &= Qx(1) = \beta\mathbf{1},\\
x(3) &= Qx(2) = \beta\mathbf{1},\\
&\vdots\\
x(k+1) &= Qx(k) = \beta\mathbf{1},\\
&\vdots
\end{aligned}
$$

- It then follows that $\beta\mathbf{1}$ is an *equilibrium point* of the distributed protocol. And this is true *for all* the possible values of the scalar β.

Distributed estimation

Linear Consensus

- In other words, if all the sensors measure the same value at the beginning, i.e. $x_i(0) = \beta \ \forall i$, that is already the arithmetic mean.
- Technically speaking, this property holds since $\mathbf{1}$ is the right *eigenvector* of Q associated to the *eigenvalue* 1, i.e.

$$Q\mathbf{1} = \mathbf{1}.$$

- This is a fundamental property of the *stochastic matrices*.

Stochastic matrices

Definition (Stochastic matrix)

A *stochastic matrix* is a matrix $Q \in \mathbb{R}^{n \times n}$ if and only if $q_{ij} \geq 0$ and $\sum_{j=0}^{n} q_{ij} = 1$, $\forall i$, i.e. the *row sum* is equal to 1.

This is the necessary property for the existence of a stable agreement (e.g. the arithmetic mean) in *linear consensus theory*.
Hence, an agreement is reached if the protocol matrix Q is a *stochastic matrix* (plus other technical requirements on matrix aperiodicity and irreducibility).

Distributed estimation

Linear Consensus

- Since the value stored in each node evolves according to the previous value stored, i.e. *discrete dynamic*, we may notice that

$$
\begin{aligned}
x(1) &= Qx(0) \\
x(2) &= Qx(1) = Q^2 x(0), \\
x(3) &= Qx(2) = Q^2 x(1) = Q^3 x(0), \\
&\vdots \\
x(k) &= Qx(k-1) = Q^k x(0), \\
&\vdots
\end{aligned}
$$

Distributed estimation

Linear Consensus

- The main property we can derive from

$$x(k) = Qx(k-1) = Q^k x(0),$$

  is that if Q^k is a matrix with *all equal rows* from some k, then after k rounds of the protocol, the system reaches an agreement, aka a *consensus*.
- Q^k is the *k-step transition matrix*, i.e. the transition matrix representing the aggregates, one shot transition from $x(0)$ to $x(k)$.
- This is trivial to show, since in that case the entries of $x(k)$, i.e. the values of the nodes after k rounds of the protocol, have all the same values.

Distributed estimation

Linear Consensus

- The fact that Q^k is a matrix with *all equal rows*, for some k, holds for a *stochastic matrix*.
- Moreover, the product of two stochastic matrices *is still a stochastic matrix*, hence if Q is a *stochastic matrix*, then Q^k is a *stochastic matrix*.
- In general nothing can be said about the reached equilibrium, i.e. what is the value of β.
- In other words, further properties should be verified to ensure that $\beta = \frac{\sum_{j=1}^{n} x_j(0)}{n}$, i.e. the arithmetic mean.

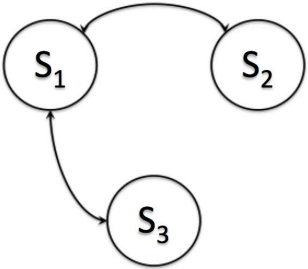
Distributed estimation

Linear Consensus

- Let us consider now the case in which there is not complete visibility among nodes, e.g. sensor S_2 does not receive the information from sensor S_3 and vice-versa.
- In this case the matrix Q changes accordingly.
- One idea can still be to compute the mean among the neighbouring nodes.
- A matrix Q built in this way is still *stochastic*.

Distributed systems

Linear Consensus



$$Q = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}$$

Figure: Sensors communication links and algebraic representation without complete visibility.

Distributed estimation

Linear Consensus

- Using the matrix product rule, for a sufficiently large k, one has for the selected Q that

$$Q^k = \begin{bmatrix} a_1 & a_2 & a_2 \\ a_1 & a_2 & a_2 \\ a_1 & a_2 & a_2 \end{bmatrix}$$

  with $a_1 > a_2$.

- Hence an agreement is reached, i.e. all the nodes will have the same quantities, but that is not the arithmetic mean, since

$$x(k) = Q^k x(0) \Rightarrow x_i(k) = a_1 x_1(0) + a_2 x_2(0) + a_2 x_3(0).$$

Stochastic matrices

To reach an *average consensus*, that is convergence towards the mean, the matrix Q should be *doubly stochastic*.

Definition (Doubly-stochastic matrix)

A stochastic matrix Q is *doubly-stochastic* iff both $\sum_{j=0}^{n} q_{ij} = 1$, $\forall i$, and $\sum_{i=0}^{n} q_{ij} = 1$, $\forall j$.

Clearly, if Q is a stochastic *symmetric* matrix, i.e., $Q = Q^T$, then it is also doubly-stochastic.
So, how we can derive a *doubly stochastic matrix* if S_2 does not see S_3?

Distributed estimation

Linear Consensus

- In this case the solution is given by

$$Q = \begin{bmatrix} x+y-1 & 1-x & 1-y \\ 1-x & x & 0 \\ 1-y & 0 & y \end{bmatrix}$$

for $0 < x < 1$ and $0 < y < 1$, with $x + y \geq 1$.

- For any value of x and y, we have a *doubly stochastic matrix*, that, for a sufficiently large k, converges to

$$Q^k = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix},$$

as desired.

Distributed estimation

Linear Consensus

- However, for different values x and y what changes is the number of messages *to reach the consensus*, i.e. the number of k to reach a matrix with all rows equal.
- One possible standard choice (independent from the number of nodes involved) is

$$q_{ij} = \begin{cases} \varepsilon & \text{if } j \text{ can communicate with } i, \\ 1 - \varepsilon d(i) & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

where q_{ij} is the element in position i and j of the matrix Q.

Distributed estimation

Linear Consensus

- For bidirectional connections, the *number of neighbours that can send and can receive* the information to and from node i is called the *node degree* and denoted with $d(i)$.
- A common usual choice is to use $\varepsilon = \frac{1}{1+\max_i d(i)}$, aka *max degree weights*.
- *WARNING!*: Notice that to compute the value of ε at least a bound on the degree of each node is needed, hence the algorithm is *partially local*.

Distributed estimation

Linear Consensus

- A different strategy for the same problem is instead given by

$$q_{ij} = \begin{cases} \frac{1}{\max(d(i),d(j))+1} & \text{if } j \text{ can communicate with } i, \\ 1 - \sum_{j=1,i\neq j}^{n} q_{ij} & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

- Notice that $\forall i$:

$$q_{ii} = 1 - \sum_{j=1,i\neq j}^{n} q_{ij} \geq 1 - \sum_{j=1,i\neq j}^{n} \frac{1}{d(i)+1} = 1 - \frac{d(i)}{d(i)+1} > 0,$$

which ensures that the matrix Q is again doubly-stochastic.

Distributed estimation

Linear Consensus

- The solution here proposed adopts the *Metropolis-Hastings* weights, which is a *local solution* and ensures *faster* convergence with respect to the previous solution.

Distributed estimation

Linear Consensus

- In both cases, the *Metropolis-Hastings* weights and the *max degree weight*, gives the following solution:

$$Q = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} & 0 \\ \frac{1}{3} & 0 & \frac{2}{3} \end{bmatrix}.$$

- Notice how this matrix is *doubly stochastic*.

Outline

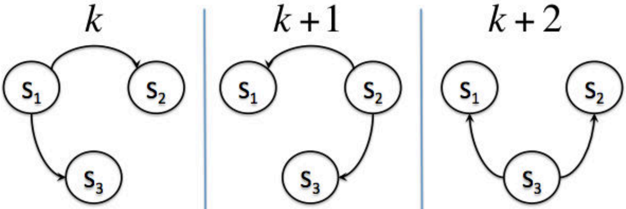
Distributed estimation

Linear Consensus

- Let us now consider a different situation in which the communication topology *changes in time*.
- This situation usually arises when two nodes cannot broadcasts their messages *simultaneously*.
- In such a case, different protocol matrices should be considered.

Distributed systems

Linear Consensus



$$Q_1 = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \quad Q_2 = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \quad Q_3 = \begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

Figure: Sensors communication links and algebraic representation with complete visibility but sequential broadcasts

Distributed estimation

Linear Consensus

- The simple choice reported in the previous figure, i.e.

$$
Q(k) = Q_1 = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}, \ Q(k+1) = Q_2 = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix},
$$

$$
Q(k+2) = Q_3 = \begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix},
$$

  computes the mean pairwise.

- At time k node S_1 broadcasts, at time $k+1$ node S_2, at time $k+2$ node S_3 and then the sequence is repeated, i.e. *round robin* scheduling.

Distributed estimation

Linear Consensus

- With the previous choice, at time k, nodes S_2 and S_3 receive the S_1 node value and then compute:

$$x_1(k+1) = x_1(k), \;\; x_i(k+1) = \frac{x_1(k) + x_i(k)}{2} \text{ for } i = 2, 3.$$

- Notice that Q_i is *stochastic* but not *doubly stochastic*.
- The stochastic matrix after a cycle of the round robin scheduling would be

$$Q = Q_3 Q_2 Q_1 = \begin{bmatrix} \frac{5}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{2} & \frac{3}{8} & \frac{1}{8} \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}$$

Distributed estimation

Linear Consensus

- Since Q is now *stochastic* but not *doubly stochastic*, a consensus would be reached, but not an *averaged consensus*.
- Since our objective is to compute the *arithmetic mean*, the solution would be to impose $Q = Q_3 Q_2 Q_1$ *doubly stochastic*, and then derive the entries of the matrices Q_i.

Distributed estimation

Linear Consensus

- A possible solution in this case is

$$Q_1 = \begin{bmatrix} 1 & 0 & 0 \\ \frac{x}{2x+1} & \frac{x+1}{2x+1} & 0 \\ \frac{x}{2x+1} & 0 & \frac{x+1}{2x+1} \end{bmatrix}, \quad Q_2 = \begin{bmatrix} \frac{1}{1+x} & \frac{x}{1+x} & 0 \\ 0 & 1 & 0 \\ 0 & \frac{x}{1+x} & \frac{1}{x+1} \end{bmatrix},$$

$$Q_3 = \begin{bmatrix} 1-x & 0 & x \\ 0 & 1-x & x \\ 0 & 0 & 1 \end{bmatrix},$$

with $0 < x < 1$.

Distributed estimation

Linear Consensus

- For example, by selecting $x = \frac{1}{2}$ one has

$$Q_1 = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{4} & \frac{3}{4} & 0 \\ \frac{1}{4} & 0 & \frac{3}{4} \end{bmatrix}, \quad Q_2 = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{3} & \frac{2}{3} \end{bmatrix},$$

$$Q_3 = \begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix}, \quad Q = Q_3 Q_2 Q_1 = \begin{bmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \end{bmatrix}.$$

- Notice that Q is *doubly stochastic* as desired, while the Q_i are *stochastic*. After some k round robin executions, $Q^k = (Q_3 Q_2 Q_1)^k$ converges to a matrix whose entries are all equal to $\frac{1}{3}$, as desired.

Distributed estimation

Linear Consensus

- The fastest convergence is instead obtained for any $x \neq 0$ and

$$Q_1 = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{6x} & \frac{1}{3x} & 0 \\ \frac{1}{6x} & 0 & \frac{1}{3x} \end{bmatrix}, \quad Q_2 = \begin{bmatrix} x & x & 0 \\ 0 & 1 & 0 \\ 0 & x & x \end{bmatrix},$$

$$Q_3 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix},$$

Distributed estimation

Linear Consensus

- Indeed, $\forall x \neq 0$ we get that matrix Q_3 has the role to substitute the value of $x_1(k)$ and $x_2(k)$ with $x_3(k)$, i.e.

$$x(k+1) = Q_3 x(k) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix}$$

while

$$Q_2 Q_1 = \begin{bmatrix} x + \frac{1}{6} & \frac{1}{3} & 0 \\ \frac{1}{6x} & \frac{1}{3x} & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}$$

that is, after the broadcasts of S_1 and S_2, S_3 has the correct value of the arithmetic mean (last row of $Q_2 Q_1$), which has to be broadcasted to the other two nodes with the last step of the round robin scheduling (i.e. matrix Q_3).

Distributed estimation

Linear Consensus

- As a consequence, only one round robin cycle is needed to reach the *average consensus*.
- This is obvious by noting that

$$Q = Q_3 Q_2 Q_1 = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix},$$

just for $k = 1$: *fastest convergence*.

Outline

Design of consensus algorithms: Examples

Problems that can be solved in a distributed sense with linear consensus algorithms:

- Node counting;
- Minimum variance estimates;
- Vehicle rendezvous;
- Least squares problems;
- Sensor calibration;
- Distributed estimation.

Outline

Consensus algorithms

Node counting

Let us consider a network with n nodes, but at start-up this number is *unknown*.

If the network satisfies the condition to reach an *average consensus*, i.e., the graph contains all the self loops and Q is doubly stochastic, then

$$\lim_{t \to +\infty} x_i(t) = \frac{1}{n} \sum_{j=1}^{n} x_j(0), \ \ \forall i = 1, \dots, n.$$

Is it possible to exploit this property?

Consensus algorithms

Node counting

If we impose that only one node, say $x_1(0)$, is equal to one and all the others equal to 0, one has

$$\lim_{t \to +\infty} x_i(t) = \frac{1}{n} \sum_{j=1}^{n} x_j(0) = \frac{1}{n}, \ \ \forall i = 1, \ldots, n.$$

Therefore, the number of nodes is simply given by

$$\frac{1}{\lim_{t \to +\infty} x_i(t)} = n, \ \ \forall i = 1, \ldots, n.$$

Outline

Consensus algorithms

Minimum variance estimates

Consider a network of n sensors all measuring the same scalar quantity

$$z_i = x + \nu_i,$$

where $\nu_i \sim \mathcal{N}(0, \sigma_i^2)$, $\forall i = 1, \dots, n$.
The *minimum variance* solution if all the sensors have the same *precision* σ_i is just the *arithmetic mean*, hence an average consensus is sufficient. However, in the general case of different sensors, we know that the solution is given by the *least squares* that is computed as

$$\hat{x}^{LS} = \sum_{i=1}^{n} \frac{\frac{z_i}{\sigma_i^2}}{\sum_{j=1}^{n} \frac{1}{\sigma_j^2}}.$$

Consensus algorithms

Minimum variance estimates

Notice that this relation can be equivalently computed as

$$\hat{x}^{LS} = \frac{\frac{1}{n}\sum_{i=1}^{n}\frac{z_i}{\sigma_i^2}}{\frac{1}{n}\sum_{j=1}^{n}\frac{1}{\sigma_j^2}}.$$

It is easy to see that this is the ratio of two average consensus algorithms. Therefore, by defining the variables $a_i(0) = \frac{z_i}{\sigma_i^2}$ and $b_i(0) = \frac{1}{\sigma_i^2}$, two parallel average consensus algorithms can be executed, i.e.

$$a(t+1) = Qa(t) \text{ and } b(t+1) = Qb(t).$$

Consensus algorithms

Minimum variance estimates

Therefore, the local estimate of the LS estimator is $\hat{x}_i^{LS}(t) = \frac{a_i(t)}{b_i(t)}$, that asymptotically converge to

$$\lim_{t \to +\infty} \hat{x}_i^{LS}(t) = \lim_{t \to +\infty} \frac{a_i(t)}{b_i(t)} = \hat{x}^{LS}, \ \forall i = 1, \dots, n.$$

Notice how the solution is completely distributed.

Outline

Consensus algorithms

Vehicle rendezvous

With the simple linear average consensus it is also possible to solve the problem of *rendezvous*, i.e. let the vehicles meet in a common point. This problem can be solved assuming that vehicles *only* use *relative distance information*.

A very simple (mono-dimensional) vehicle kinematic can be described by

$$x_i(t+1) = x_i(t) + u_i(t).$$

Consensus algorithms

Vehicle rendezvous

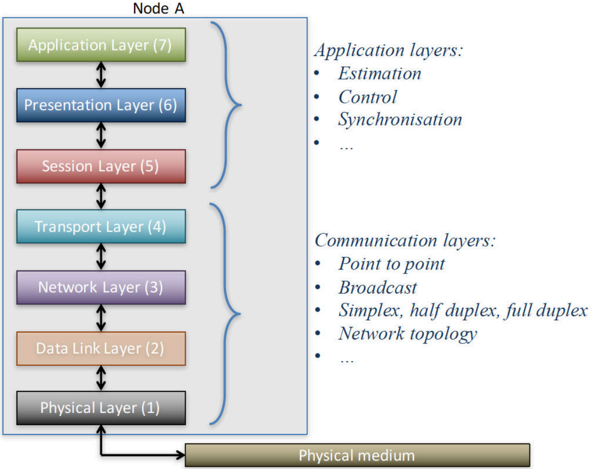
It is then easy to see that a control law such as

$$u_i(t) = \sum_{j=1}^{n} q_{ij}(x_j(t) - x_i(t)),$$

with properly chosen weights to guarantee average consensus leads to vehicle rendezvous.

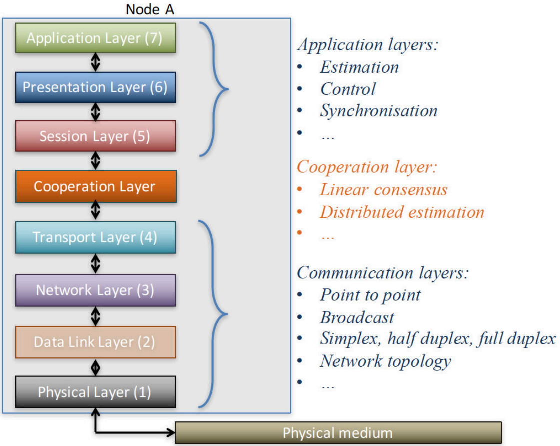
Outline

OSI Model



Convenient renaming of the OSI model layers.

OSI Model



The consensus protocol can be seen as a *cooperation layer*.