Getting multi-agent systems to cooperate

Luca Schenato
University of Padova
ECC 2014
Outline

- Motivations and target applications
- Challenges
- The consensus algorithm
- Application of consensus
- Conclusions and open vistas
Outline

- Motivations and target applications
- Challenges
- The consensus algorithm
- Application of consensus
- Conclusions and open vistas
Networked Control Systems

Physically distributed dynamical systems interconnected by a communication network
Research lines

- **Research line 1: multi-agent systems:**
  - Consensus algorithms
  - Distributed estimation
  - Distributed optimization

- **Research line 2: control subject to communication constraints:**
  - Packet loss
  - Random delay
  - Quantization
Target applications: MAgIC Lab. at University of Padova

- Wireless Sensor Actuator Networks
- Smart Camera Networks
- Robotic Networks
- Smart Energy Grids

MAgIC
Multi Agent Intelligent Control
Joint work with

Colleagues at Univ. of Padova

Ruggero Carli  Angelo Cenedese  Alessandro Chiuso  Gianluigi Pillonetto  Sandro Zampieri

Former/current students:

Andrea Carron  Marco Todescato  Simone Del Favero

International collaborators:

Sinan Yildirim  Ege Univ., Turkey  Fabio Fagnani  Turin Politech, Italy  Lara Brinon-Arranz  IST, Portugal

Damiano Varagnolo  Univ. of Lulea, Sweden  Saverio Bolognani  MIT, USA  Filippo Zanella  Sellf Inc.

Alexandra von Meier  UC. Berkeley, USA  Reza Argandeh  CIEE, Berkeley, USA  Kameshwar Poolla  UC. Berkeley, USA
Acknowledgements

Highly-complex and networked control systems
EU Network of Excellence: 2010-2014

Co-design for Networked Control Systems
EU STREP: 2008-2012

Wireless Sensor Networks for city-wide Ambient Intelligence
National Research Project: 2008-2011

Distributed Systems for Energetic and Environmental Monitoring
National Industrial Project: 2010-2011
Outline

- Motivations and target applications
- Challenges
  - The consensus algorithm
  - Application of consensus
- Conclusions and open vistas
Challenges

- **Unreliable (wireless) communication:**
  - Random delay, packet loss, limited communication range

- **Scalability:**
  - Complexity (CPU, memory, communication) per agent must be constant

- **Robustness:**
  - Mild performance degradation when local failures

- **Architecture:**
  - Centralized vs hierarchical vs distributed vs decentralized
  - Cooperative vs competitive
Challenges: a personal experience

- **Prototyping time**
  - Leader-based/hierarchical algorithms too complex to write

- **Debugging time**
  - Few LEDs for visual inspection
  - Ex-post analysis of dozens of agent data logs after a failure

- **Rapid peer-to-peer communication**
  - Wi-Fi, bluetooth, zigbee not suitable for peer-to-peer

Need for simple asynchronous peer-to-peer algorithms
Some working complex systems

INTERNET

Cell phones networks
A leading paradigm: ISO layers with few primitives

ISO Protocol Model

Application Layer
Transport Layer
Network Layer
Link Layer
Physical Layer

TCP/IP Stack
APPLICATION
TCP/UDP
IP
Link Layer
Physical Layer

Message
Segment
Datagram
Frame

Application layer
Communication layers
Multi-agent systems: an ISO-like paradigm?

What should be the right ISO-model? Need to seamlessly integrate:
- Communication network(s)
- Sensing and control
- Physical constraints (conservation mass/energy)
- Markets

Smart Power Grids

Intelligent transportation
ISO for multi-agent systems

Application layer

- Time-synch
- Sensor calibration
- Map building
- ???

Communication layer

- Point-to-point
- Broadcast
- Multi-cast
- ???
Consensus algorithm: a primitive for cooperation

Application layer:
- Time-synch
- Sensor calibration
- Map building

Cooperation layer:
- Average consensus
- Consensus
- Map building

Communication layer:
- Point-to-point
- Broadcast
- Multi-cast
Outline

- Motivations and target applications
- Challenges
- The consensus algorithm
- Application of consensus
- Conclusions and open vistas
The consensus problem

Main idea
- Having a set of agents to agree upon a certain value (usually global function) using only local information exchange (local interaction)

Also known as:
- Agreement problem (economics, social networks)
- Load balancing (Computer Science & communications)
- Synchronization (statistical mechanics)
- Rendezvous and flocking (robotics)

Old problem: Markov Chains (60’s), Load balancing (’70), Distributed decision making (80’s), flocking (00’s)
Multi-agent modeling

- Network of
  - N agents
  - Communication graph: $G = (\mathcal{V}, \mathcal{E})$
  - i-th node neighbors: $\mathcal{N}_i$
  - Local variable: node $i$ store $x_i$

$\mathcal{V} = \{1, 2, 3, 4, 5, 6\}$
$\mathcal{E} = \{(1, 1), (1, 2), (2, 1), (1, 3), \ldots\}$
$\mathcal{N}_1 = \{2, 3, 4\}$
Recursive Distributed Algorithms

DEFINITION: Recursive Distributed Algorithm consistent with the graph $G$:

Any recursive algorithm where the $i$-th node’s update law depends only on the local variables of $i$ and its neighbors

$$x_i(t + 1) = f(x_i(t), \{x_j(t)\}_{j \in \mathcal{N}_i}, t)$$
**DEFINITION:**
A Recursive Distributed Algorithm consistent with the graph G is said to **asymptotically achieve consensus** if

\[ x_i(t) \xrightarrow{} \alpha, \ \forall i \]

**DEFINITION:**
A Recursive Distributed Algorithm consistent with the graph G is said to **asymptotically achieve average consensus** if

\[ x_i(t) \xrightarrow{} \frac{1}{N} \sum_i x_i(0), \ \forall i \]
A robotics example: the rendezvous problem

Receiving node:
\[ x_4(t + 1) = \frac{1}{2} x_3(t) + \frac{1}{4} x_1(t) + \frac{1}{4} x_2(t) \]

Other nodes:
\[ x_1(t + 1) = x_1(t) \]
\[ x_2(t + 1) = x_2(t) \]
\[ x_3(t + 1) = x_3(t) \]

\[ x_i(t + 1) = x_i(t) + u_i(t) \]
A robotics example: the rendezvous problem

Receiving node:
\[ x_2(t + 1) = \frac{1}{2} x_2(t) + \frac{1}{2} x_4(t) \]
\[ x_3(t + 1) = \frac{1}{2} x_3(t) + \frac{1}{2} x_4(t) \]

Other nodes:
\[ x_1(t + 1) = x_1(t) \]
\[ x_4(t + 1) = x_4(t) \]
A robotics example: the rendezvous problem

Receiving node:

\[
\begin{align*}
    x_1(t + 1) &= x_4(t + 1) = \frac{1}{2} x_1(t) + \frac{1}{2} x_4(t) \\
    x_2(t + 1) &= x_3(t + 1) = \frac{1}{2} x_2(t) + \frac{1}{2} x_3(t)
\end{align*}
\]
The linear consensus algorithm

\[
x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}, \quad 1 = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}
\]

\[
x_i(t+1) = p_{ii}x_i(t) + \sum_{j \in N_i} p_{ij}x_j \\
x_i(t+1) = x_i(t) + \sum_{j \in N_i} p_{ij}(x_j - x_i) \\
x(t+1) = P(t)x(t) \\
x(t+1) = x(t) + (P(t) - I)K(t)x(t)
\]

PROPERTIES OF \( P(t) \) (Stochastic Matrix)
- Consistent with the graph: \( P(t) \sim G \) \( (P_{ij}(t) = 0 \text{ if } (j, i) \notin \mathcal{E}) \)
- Component-wise non-negative: \( P_{ij}(t) \geq 0 \)
- Row-sum unitary: \( P(t)1 = 1 \)

\( P(t) \) doubly stochastic if also column-sum unitary: \( 1^TP(t) = 1^T \)
Constant matrix $P$

**Synchronous communication:**
At each time all nodes communicate according to the communication graph and update their local variables

$$P(t) = P$$

$$P = \begin{bmatrix}
\frac{3}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & 0 & 0 \\
\frac{1}{6} & \frac{3}{6} & \frac{1}{6} & 0 & 0 & \frac{1}{6} \\
\frac{1}{6} & \frac{1}{6} & \frac{3}{6} & 0 & \frac{1}{6} & 0 \\
\frac{1}{6} & 0 & 0 & \frac{3}{6} & \frac{1}{6} & \frac{1}{6} \\
0 & 0 & \frac{1}{6} & \frac{1}{6} & \frac{4}{6} & 0 \\
0 & \frac{1}{6} & 0 & \frac{1}{6} & 0 & \frac{4}{6} \\
\end{bmatrix}$$

(Laplacian weights)
Time varying $P(t)$: broadcast

Broadcast communication:
At each time one node wakes up and broadcasts its information to all its neighbors

$P(t) \in \{P_1, P_2, P_3, P_4, P_5, P_6\}$

$$P_4 = \begin{bmatrix}
3/4 & 0 & 0 & 1/4 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1/4 & 3/4 & 0 \\
0 & 0 & 0 & 1/4 & 0 & 3/4
\end{bmatrix}$$
Symmetric gossip communication:
At each time one node wakes up and chooses one of its neighbors. These two nodes exchange their local variables.

\[ P(t) \in \{P_{(12)}, P_{(14)}, \ldots, P_{(46)}\} \]

\[ P_{(14)} = \begin{bmatrix}
    1/2 & 0 & 0 & 1/2 & 0 & 0 \\
    0 & 1 & 0 & 0 & 0 & 0 \\
    0 & 0 & 1 & 0 & 0 & 0 \\
    1/2 & 0 & 0 & 1/2 & 0 & 0 \\
    0 & 0 & 0 & 0 & 1 & 0 \\
    0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix} \]
Asynchronous consensus: convergence

- **Standard Consensus (Broadcast)**
  - Graph rooted on average
  - Self-loops, i.e. P(t) with positive diagonal
  - P(t) row-stochastic

\[ x_i(t) \longrightarrow \sum_{j=1}^{N} \alpha_j x_j(0), \quad \alpha_j \geq 0, \sum_j \alpha_j = 1 \]

- **Average Consensus (Gossip)**
  - Graph connected on average
  - Self-loops, i.e. P(t) with positive diagonal
  - P(t) doubly stochastic \( 1^T P(t) = 1^T \)

\[ x_i(t) \longrightarrow \sum_{j=1}^{N} \frac{1}{N} x_j(0) \]
Convergence for time-varying communication
Asynchronous consensus: communication burden

- **Broadcast-based Consensus**
  - Achieves consensus
  - $N_i$ updates per 1 sent message
  - No ACK message required

- **Gossip-based Consensus**
  - Achieves average consensus
  - 2 updates per (at least) 3 sent messages
  - Non-trivial communication protocol
Average consensus:
the (broadcast) ratio consensus

**Standard**

Transmitter node
\[
x_4(t + 1) = x_4(t)
\]

Receiver nodes
\[
x_1(t + 1) = \frac{3}{4} x_1(t) + \frac{1}{4} x_4(t)
\]
\[
x_5(t + 1) = \frac{3}{4} x_5(t) + \frac{1}{4} x_4(t)
\]
\[
x_6(t + 1) = \frac{3}{4} x_6(t) + \frac{1}{4} x_4(t)
\]

Other nodes:
\[
x_2(t + 1) = x_2(t)
\]
\[
x_3(t + 1) = x_3(t)
\]

**Ratio**

Transmitter node
\[
x_4(t + 1) = (1 - \frac{3}{4}) x_4(t)
\]

Receiver nodes
\[
x_1(t + 1) = x_1(t) + \frac{1}{4} x_4(t)
\]
\[
x_5(t + 1) = x_5(t) + \frac{1}{4} x_4(t)
\]
\[
x_6(t + 1) = x_6(t) + \frac{1}{4} x_4(t)
\]

Other nodes:
\[
x_2(t + 1) = x_2(t)
\]
\[
x_3(t + 1) = x_3(t)
\]

**Row stochastic**

\[
P(t) = \begin{bmatrix}
\frac{3}{4} & 0 & 0 & 1/4 & 0 & 0
0 & 1 & 0 & 0 & 0 & 0
0 & 0 & 1 & 0 & 0 & 0
0 & 0 & 0 & 1 & 0 & 0
0 & 0 & 0 & 1/4 & 3/4 & 0
0 & 0 & 0 & 1/4 & 0 & 3/4
\end{bmatrix}
\]

**Column stochastic**

\[
Q(t) = \begin{bmatrix}
1 & 0 & 0 & 1/4 & 0 & 0
0 & 1 & 0 & 0 & 0 & 0
0 & 0 & 1 & 0 & 0 & 0
0 & 0 & 0 & 1/4 & 0 & 0
0 & 0 & 0 & 1/4 & 1 & 0
0 & 0 & 0 & 1/4 & 0 & 1
\end{bmatrix}
\]
Average consensus: the ratio consensus

\[ P(t) = \begin{bmatrix}
    3/4 & 0 & 0 & 1/4 & 0 & 0 \\
    0 & 1 & 0 & 0 & 0 & 0 \\
    0 & 0 & 1 & 0 & 0 & 0 \\
    0 & 0 & 0 & 1 & 0 & 0 \\
    0 & 0 & 0 & 1/4 & 3/4 & 0 \\
    0 & 0 & 0 & 1/4 & 0 & 3/4 \\
\end{bmatrix} \]

\[ x(t + 1) = P(t)x(t) \]

\[ x_i(t) \rightarrow \sum_j \alpha_j x_j(0), \forall i \]

\[ \alpha_i > 0, \sum_i \alpha_i = 1 \]

\[ P(t) = \begin{bmatrix}
    1 & 0 & 0 & 1/4 & 0 & 0 \\
    0 & 1 & 0 & 0 & 0 & 0 \\
    0 & 0 & 1 & 0 & 0 & 0 \\
    0 & 0 & 0 & 1/4 & 0 & 0 \\
    0 & 0 & 0 & 1/4 & 1 & 0 \\
    0 & 0 & 0 & 1/4 & 0 & 1 \\
\end{bmatrix} \]

\[ x(t + 1) = Q(t)x(t) \]

\[ x_i(t) \rightarrow \beta_i(t) \sum_j x_j(0), \forall i \]

\[ \beta_i(t) > 0, \sum_i \beta_i(t) = 1, \forall t \]

\[ y(t + 1) = Q(t)y(t), \quad y(0) = 1 \]

\[ z_i(t) := \frac{x_i(t)}{y_i(t)} \quad \rightarrow \quad \frac{\beta_i(t) \sum_j x_j(0)}{\beta_i(t) \sum_j 1} \]

\[ z_i(t) \rightarrow \sum_j \frac{1}{N} x_j(0), \forall i \]

- D. Kempe, A. Dobra, and J. Gehrke, 2003
- M. Alighanbari and J. How, 2008
Realistic scenarios

Ideal scenario

Collisions

Packet losses
Packet loss: Broadcast consensus

**Standard**

Transmitter node
\[ x_4(t + 1) = x_4(t) \]

Receiver nodes
\[
\begin{align*}
    x_1(t + 1) &= x_1(t) \\
    x_5(t + 1) &= \frac{3}{4}x_5(t) + \frac{1}{4}x_4(t) \\
    x_6(t + 1) &= \frac{3}{4}x_6(t) + \frac{1}{4}x_4(t)
\end{align*}
\]

Other nodes:
\[
\begin{align*}
    x_2(t + 1) &= x_2(t) \\
    x_3(t + 1) &= x_3(t)
\end{align*}
\]

**Ratio**

Transmitter node
\[ x_4(t + 1) = (1 - \frac{3}{4})x_4(t) \]

Receiver nodes
\[
\begin{align*}
    x_1(t + 1) &= x_1(t) \\
    x_5(t + 1) &= x_5(t) + \frac{1}{4}x_4(t) \\
    x_6(t + 1) &= x_6(t) + \frac{1}{4}x_4(t)
\end{align*}
\]

Other nodes:
\[
\begin{align*}
    x_2(t + 1) &= x_2(t) \\
    x_3(t + 1) &= x_3(t)
\end{align*}
\]

\[ P(t) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/4 & 3/4 & 0 \\ 0 & 0 & 0 & 1/4 & 0 & 3/4 \end{bmatrix} \]

\[ Q(t) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/4 & 0 & 0 \\ 0 & 0 & 0 & 1/4 & 1 & 0 \\ 0 & 0 & 0 & 1/4 & 0 & 1 \end{bmatrix} \]

**Row stochastic**
\[ x_i(t) \rightarrow \sum_j \alpha_j x_j(0), \forall i \]

**Column sub-stochastic**
\[ x_i(t) \rightarrow 0, \forall i \]
Packet losses:
symmetric gossip consensus

Gossip nodes
\[ x_1(t + 1) = x_1(t) \]
\[ x_4(t + 1) = \frac{1}{2} x_4(t) + \frac{1}{2} x_1(t) \]

Other nodes
\[ x_2(t + 1) = x_2(t) \]
\[ x_3(t + 1) = x_3(t) \]
\[ x_5(t + 1) = x_5(t) \]
\[ x_6(t + 1) = x_6(t) \]

\[ P(t) = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1/4 & 3/4 & 0 \\
0 & 0 & 0 & 1/4 & 0 & 3/4 \\
\end{bmatrix} \]

Row stochastic
\[ x_i(t) \rightarrow \sum_j \alpha_j x_j(0), \forall i \]
Asynchronous consensus: packet loss and random delay

- **Standard Consensus (broadcast)**
  - Guaranteed (slower) convergence
  \[ x_i(t) \rightarrow \sum_{j=1}^{N} \alpha_j x_j(0) \]

- **Average Consensus (gossip)**
  - Guaranteed (slower) convergence, but loss of average
  - Under randomized communication: \( N \rightarrow \infty \Rightarrow \alpha_i \rightarrow \frac{1}{N} \)
  (Fagnani-Zampieri 2009, Frasca-Hendrickx 2013)

- **Ratio Consensus (broadcast)**
  - No convergence

- **Robust Ratio Consensus (broadcast)**
  - Guaranteed average consensus
  \[ x_i(t) \rightarrow \sum_{j=1}^{N} \frac{1}{N} x_j(0) \]
  - Additional local variables required
  (Domínguez-Garcis-Hadjicostis-Vaidya, 2014)
Consensus algorithm: a primitive for multi-agent systems

- **Application layer**
  - Time-synch
  - Sensor calibration
  - Distributed optimization
  - ???

- **Cooperation layer**
  - Average consensus
  - Consensus
  - ???

- **Communication layer**
  - Point-to-point
  - Broadcast
  - Multi-cast
  - ???

Robust asynchronous broadcast-based and relatively simple implementations available
Motivations and target applications

Challenges

The consensus algorithm

Application of consensus

Conclusions and open vistas
Consensus-based applications

Wireless Sensor Actuator Networks
- Sensor Calibration
- RF indoor tracking
- Clock Synchronization
- Cardinality estimation

Smart Camera Networks
- Perimeter patrolling

Robotic Networks
- Rendez-vous
- Map building
- Localization
- Source-seeking

Smart Energy Grids
- Multi-area state estimation
Sensor calibration issues in RF-based localization

\[ \Gamma_{ij} - \Gamma_{ji} \approx \text{const.} \]

Systematic calibration errors

\[ \Gamma_{ij} = g(x_i, x_j) + o_i \]
\[ \Gamma_{ji} = g(x_j, x_i) + o_j \]
\[ g(x_i, x_j) = g(x_j, x_i) \]
WSN sensor calibration

Ideally:
- Estimate $o_i : \hat{o}_i$
- Use $\hat{o}_i$ to compensate the offset: $o_i - \hat{o}_i = 0$

Calibrated measurement

$$\hat{\Gamma}_{ij} = g(x_i, x_j) + o_i - \hat{o}_i$$

$$\Gamma_{ij} - \Gamma_{ji} = o_i - o_j$$

What we propose is:

$$o_i - \hat{o}_i = \alpha, \quad \alpha \approx 0, \quad \forall i$$

All nodes overestimate or underestimate the distance similarly. The errors, in the triangulation process, cancel out partially.
Define \( x(t) := o_i - \hat{o}_i(t) \) we want \( x(t) \to \alpha \)

Recalling:

\[
x_i(t+1) = x_i(t) + \sum_{j \in N_i} p_{ij} (x_j - x_i)
\]

\[
\hat{o}_i(0) = 0
\]

\[
\Gamma^{ij} = g(x_i, x_j) + o_i \quad \Gamma^{ji} = g(x_j, x_i) + o_j
\]

\[
g(x_i, x_j) = g(x_j, x_j)
\]

\[
\hat{o}_i(t+1) = \hat{o}_i(t) - \sum_j p_{ij} \left( \Gamma^{ij} - \Gamma^{ji} - \hat{o}_i(t) + \hat{o}_j(t) \right)
\]

update equation

\[
\hat{o}_i(t) \to o_i - \sum_i \alpha_i o_i \approx o_i - \frac{1}{N} \sum_i o_i \approx o_i
\]

Steady state
Experimental Testbed

25 Tmote-Sky nodes with Chipcon CC2420 RF transceiver randomly placed inside a single conference room

Network topology and nodes displacement:
Edge if packet loss probability <25%
Experimental results: Broadcast consensus

Links divided into 2 categories:
- Training links (black)
- Validation links (gray)

Error distribution

![Error distribution graph](image)

Oscillations

![Oscillations graph](image)

Number of consensus iterations

![Number of consensus iterations graph](image)
Estimation from noisy relative measurements

$$\Gamma^{ij} = g(x_i, x_j) + o_i + v_{ij}$$
$$\Gamma^{ji} = g(x_j, x_i) + o_j + v_{ji}$$
$$g(x_i, x_j) = g(x_j, x_i)$$

\[
\min_{\hat{o}_1, \ldots, \hat{o}_N} \sum_{(i,j)\in E} |\Gamma^{ij} - \Gamma^{ji} + \hat{o}_i - \hat{o}_j|^2
\]

- **Synchronous implementations:**
  - Barooah 2007

- **Asynchronous implementations:**
  - P. Barooah and J. P. Hespanha, 2005
  - A. Giridhar and P. R. Kumar, 2006
  - N. M. Freris and A. Zouzias, 2012
  - C. Ravazzi, P. Frasca, H. Ishii, and R. Tempo, 2013

- **Asynchronous implementation robust to packet losses and random delays**
  - M. Todescato, A. Carron, R. Carli, L. Schenato, 2014
Clock Synchronization in WSN

Low Power TDMA communication for battery powered nodes
Clock Synchronization: cascade consensus

Hardware clocks
\[ \tau_i(t) = \alpha_i t + \beta_i \quad i = 1, \ldots, N \]

Virtual reference clock
\[ \tau^*(t) = \alpha^* t + \beta^* \]

Software clock
\[ \hat{\tau}_j(t) = \hat{\alpha}_j \tau_i + \hat{\beta}_j \quad i = 1, \ldots, N \]

Goal:
find \((\hat{\alpha}_j, \hat{\beta}_j)\) such that
\[ \lim_{t \to \infty} \hat{\tau}_i(t) = \tau^*(t), \forall i = 1, \ldots, N \]

\[ \hat{\tau}_j(t) = \hat{\alpha}_j \alpha_i t + \hat{\alpha}_i \beta_j + \hat{\beta}_j \]

\[ \alpha^* \]

\[ \beta^* \]
Clock Synchronization: cascade consensus

\[
\hat{\tau}_i(t) = \hat{\alpha}_j \alpha_j t + \hat{\alpha}_i \beta_i + o_j
\]

\[
x_j^\alpha(t^+) = \frac{1}{2} x_j^\alpha(t) + \frac{1}{2} x_i^\alpha(t)
\]

\[
\hat{\alpha}_j(t^+) = \frac{1}{2} \hat{\alpha}_j(t) + \frac{1}{2} \hat{\alpha}_i(t) \frac{\alpha_i}{\alpha_j}
\]

Drift compensation

\[
\hat{\alpha}_j(t^+) = \frac{1}{2} \hat{\alpha}_j(t) + \frac{1}{2} \hat{\alpha}_i(t) \frac{\tau_i(t_2) - \tau_i(t_1)}{\tau_j(t_2) - \tau_j(t_1)}
\]

Offset compensation

\[
\hat{o}_i^+ = \hat{o}_i - \frac{1}{2}(\hat{\tau}_i - \hat{\tau}_j)
\]

- Solis, Borkar, Kumar, 2006
- Sommer, Wattenhofer, 2009
- Fiorentin, Schenato 2011
- Liao, Barooha 2013
Clock Synchronization: PI consensus

**Virtual reference clock**
\[ \tau^*(t) = \alpha^* t + \beta^* \]

**Software clock**
\[ \hat{\tau}_j(t) = \hat{\alpha}_j \tau_i + \hat{\beta}_j \quad i = 1, \ldots, N \]

**Hardware clocks**
\[ \tau_i(t) = \alpha_i t + \beta_i \quad i = 1, \ldots, N \]

**PI consensus:**
\[ \hat{\omega}_i(t^+) = \hat{\omega}_i(t) - 1(\hat{\tau}_i(t) - \hat{\tau}_j(t)) \]
\[ \hat{\alpha}_i(t^+) = \hat{\alpha}_i(t) - K_I(\hat{\tau}_i(t) - \hat{\tau}_j(t)) \]

**Cascade consensus:**
\[ \hat{\omega}_i^- = \hat{\omega}_i - \frac{1}{2}(\hat{\tau}_i - \hat{\tau}_j) \]
\[ \hat{\alpha}_j^- = \frac{1}{2} \hat{\alpha}_j + \frac{1}{2} \hat{\alpha}_i \frac{\tau_i(t_2) - \tau_i(t_1)}{\tau_j(t_2) - \tau_j(t_1)} \]

- Carli, Chiuso, Schenato, Zampieri 2006
- Yildirim, Carli, Schenato, 2014
Clock Synch in WSN: experiments

**Actual code**

```c
void synchronize(TimeSyncMsg *msg)
{
    int32_t timeError;
    float newSkew = skew;

    /* calculate offset difference */
    timeError = msg->localTime - msg->globalTime - timeError;

    /* adjust the speed of the logical clock */
    if (timeError < E_MIN && timeError > -E_MAX) {
        /* turn on integrator */
        /* calculate adaptive alpha */
        if (lastError != 0 && lastError != timeError) {
            currentAlpha *= (float)lastError/(float)(lastError - timeError);
        }
        currentAlpha = fabs(currentAlpha);
    }
    if (currentAlpha > ALPHA_MAX) currentAlpha = ALPHA_MAX;
    /* adjust rate multiplier */
    newSkew += currentAlpha * (float)timeError;
}

lastError = timeError;
/* update logical clock parameters */
atomic(
    skew = newSkew;
    clock = msg->globalTime;
    lastUpdate = msg->localTime;
}
```

**Complexity**

- **CPU Overhead (ticks)**: FTSP ≈ 5440, PulseSync ≈ 5440, GTSP ≈ 5610, PISync ≈ 145
- **Message Length (bytes)**: 9, 9, 9, 4-9
- **Main Memory Overhead (bytes)**: 52, 52, 64*|N| + 12, 16
- **Flash Memory Requirements (bytes)**: 18000, 17856, 22092, 15432

- x50 faster
- x4-20 less RAM memory
Clock Synch in WSN: video

Courtesy of Sinan Yildirim, Ege University, Turkey
Map-building in robotic networks

Parametric Model:
\[ g(x) = \sum_{m=1}^{M} \theta_m g_m(x) \]

Noisy data:
\[ \{(x_i, y_i)\}_{i=1}^{N} \]
\[ y_i = g(x_i) + v_i \]

Scenarios
- Each robot collects local data
- Local communication with robot
- Patrolled area dynamically change

Goal:
\[ \min_\theta \sum_i v_i^2 \]
Map-building as least-squares regression

- **Model class:**
  \[
  g(x) = \sum_{m=1}^{M} \theta_m g_m(x)
  \]

- **Noisy measurements:**
  \[
  y_i = \sum_{m=1}^{M} \theta_m g_m(x_i) + v_i, \quad i = 1, \ldots, N
  \]
  \[
  \begin{bmatrix}
  y(x_1) \\
  y(x_2) \\
  \vdots \\
  y(x_N)
  \end{bmatrix} =
  \begin{bmatrix}
  g_1(x_1) & \cdots & g_M(x_1) \\
  g_1(x_2) & \cdots & g_M(x_2) \\
  \vdots & \ddots & \vdots \\
  g_1(x_N) & \cdots & g_M(x_N)
  \end{bmatrix}
  \begin{bmatrix}
  \theta_1 \\
  \vdots \\
  \theta_M
  \end{bmatrix}
  +
  \begin{bmatrix}
  v_1 \\
  \vdots \\
  v_N
  \end{bmatrix}
  \]

  \[y = G\theta + v\]

- **Goal:** minimize sum of squares of residues
  \[
  \hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{N} v_i^2
  \]

  \[
  \hat{\theta} = \left(\sum_{i=1}^{N} G_i G_i^T\right)^{-1} \left(\sum_{i=1}^{N} G_i y_i\right)
  \]

  \[
  = \left(\frac{1}{N} \sum_{i=1}^{N} G_i G_i^T\right)^{-1} \left(\frac{1}{N} \sum_{i=1}^{N} G_i y_i\right)
  \]

- Xiao-Boyd-Lall, 2005
- Bolognani-Del Favero-Schenato-Varagnolo, 2010
**Consensus-based Map-building: gossip communication**

**ALGORITHM:**

1) Initialize statistics:
\[ Z^i_0 = 0 \in R^{M \times M} \]
\[ z^i_0 = 0 \in R^M \]

2) Collect data and build local statistics:
\[ Z^i_{t+1} = Z^i_t + G^i_t G^i_t T \]
\[ z^i_{t+1} = z^i_t + G^i_t y^i_t \]

3) Choose neighbor \( j \) and do gossip consensus:
\[ Z^j_{t+1} = Z^{i}_{t+1} = \frac{1}{2} Z^i_t + \frac{1}{2} Z^j_t \]
\[ z^j_{t+1} = z^{i}_{t+1} = \frac{1}{2} z^i_t + \frac{1}{2} z^j_t \]

4) Estimate map:
\[ \hat{\theta}^i_t = (Z^i_t)^{-1} z^i_t \]

5) Repeat steps 2,3,4 (non necessarily in order)
Consensus-based map-building: robust broadcast ratio consensus

Courtesy of Damiano Varagnolo, University of Lulea, Sweden
Cooperative distributed optimization

Agents cooperate to find the minimizer of the network cost:

\[
f(x) = \frac{1}{N} \sum_{i=1}^{N} f_i(x), \quad x^* = \arg\min_x f(x)
\]

- **Global estimation:**
  - Each node wants a copy of the global minimizer \( \hat{x}_i = x^* \)
  - Machine learning, map building, ....

- **Local estimation:**
  - \( f_i(x) = f_i(x_i, \{x_j\}_{j \in \mathcal{N}_i}) \) each node just wants \( \hat{x}_i = x_i^* \)
  - Calibration, localization, ....

\( N \): number of agents
\( n \): state dimension \((x \in \mathbb{R}^n)\)
On going work:
Newton-Raphson Consensus

- Distributed optimization very popular research area:
  - Augmented Lagrangians (ADMM)
  - Sub-gradient methods
  - ...
- Asynchronous and robust distributed optimization still very open and practically relevant
- Our recent effort in merging Newton-Raphson and consensus ideas together
- HYCON2 Workshop on Distributed Optimization in Large Networks and its Applications, ECC-2013 (slides available on-line)
Outline

- Motivations and target applications
- Challenges
- The consensus algorithm
- Application of consensus
- Conclusions and open vistas
Conclusions

- Consensus as a building block for cooperative multi-agent applications

- Effort is in casting general problems as consensus

- Time-varying higher order consensus is still an open problem
  - PI consensus (clock synch)
  - PD consensus (fast consensus, diffusive algorithms)
  - PID (?)

- Self-tuning: adaptive tuning of parameters/gains in distributed algorithms
Open vistas (1)

- **Architecture:** Multi-agent/complex systems still an open challenge
Open vistas (2)

- **Computation**: Asynchronous distributed algorithms robust to unreliable communication

[Parallel computing (old paradigm)](image1)

[Cloud computing (new paradigm)](image2)
Open vistas (3)

- **Data Tsunami (≠Big data):** most data is time-series.
  - Time and causality must be treated differently than usually done in machine learning
  - Cooperative multi-agent algorithms will be a necessity
Thank you

URL: http://automatica.dei.unipd.it/people/schenato.html
References (1)

- **Consensus:**

- **Sensor calibration:**
  - M. Todescato, A. Carron, R. Carli, L. Schenato. *Distributed Localization from Relative Noisy Measurements: a Robust Gradient Based Approach.* *IEEE Conference on Decision and Control (CDC14), submitted*

- **Clock synchronization:**
References (2)

- **Map Building:**

- **Distributed optimization:**