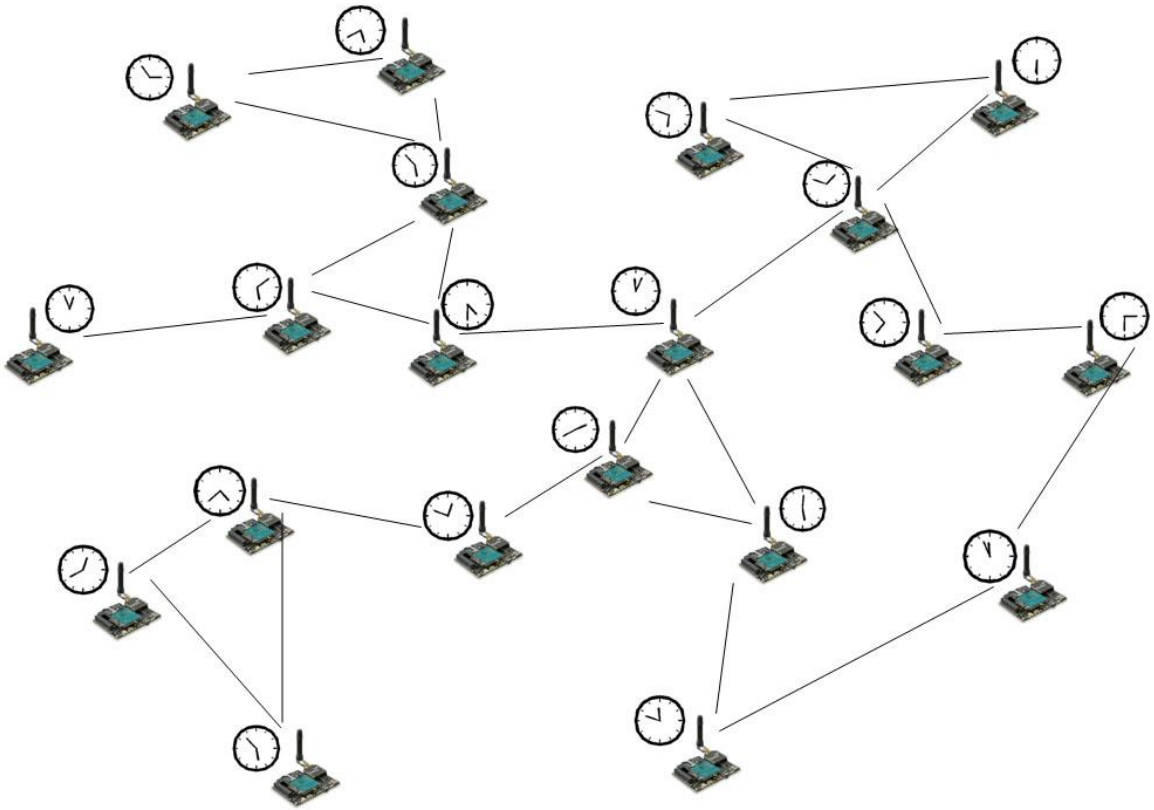
Clock Synchronization in WSN: from Traditional Estimation Theory to Distributed Signal Processing

Yik-Chung WU

The University of Hong Kong

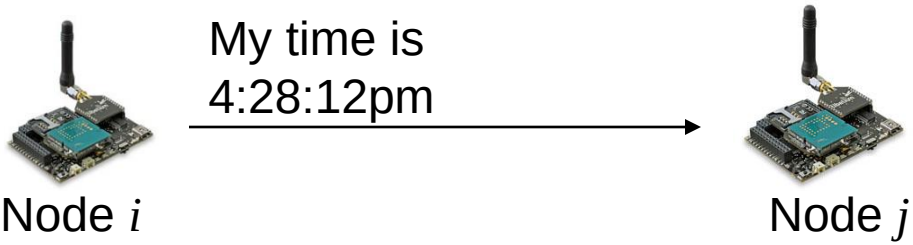
Email: ycwu@eee.hku.hk, Webpage: www.eee.hku.hk/~ycwu



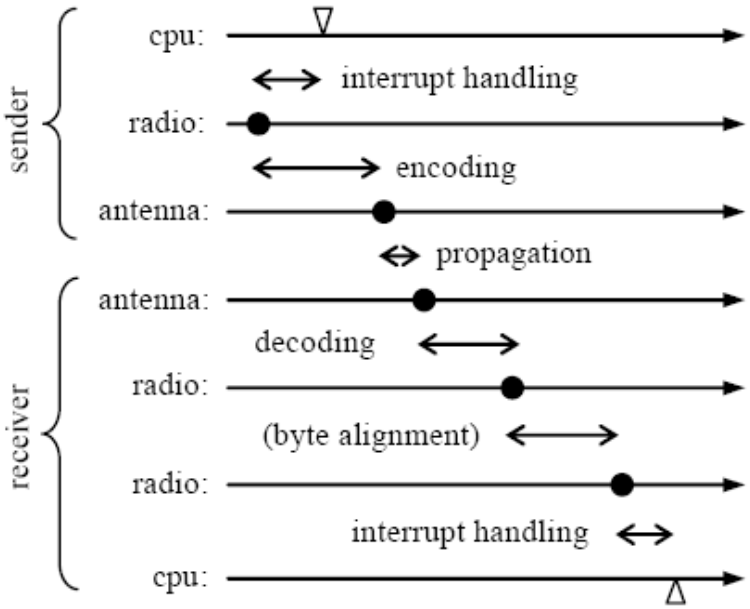
- Applications require clock synchronization
  - Event detection
  - Data Fusion
  - Sleep and wake-up cycle for power management
  - TDD transmission schedule

Challenges?

- Ideal case

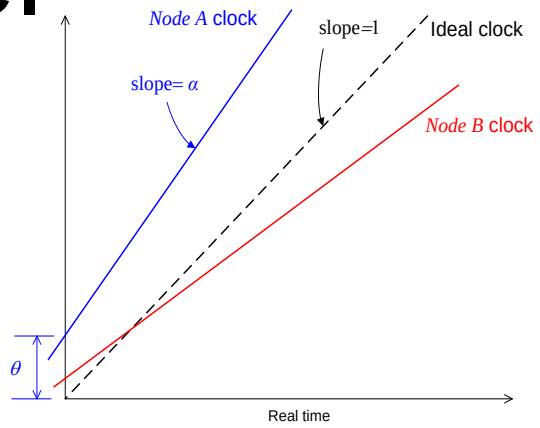


- In reality delays exist
  - The delay between radio chip interpret and the CPU responding
  - The time for radio chip to transform the message to EM wave
  - The time for converting the received EM wave into the original message
  - Finally signal to CPU reception is completed



System model

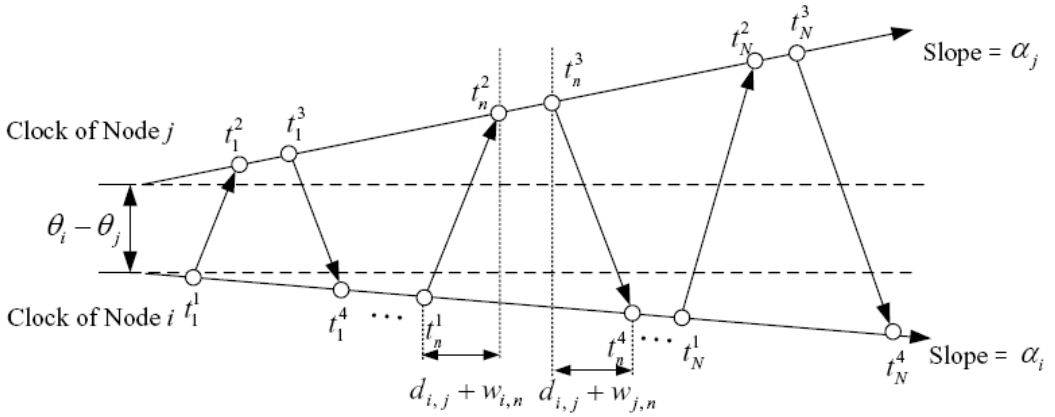
- Clock model: $c_i(t) = \alpha_i t + \theta_i$



- Two-way message exchange is used to establish the clock relationship between two nodes:

$$\text{real\_time}\big|_{t_n^2} = \text{real\_time}\big|_{t_n^1} + d_{i,j} + w_{j,n}$$

$$\text{real\_time}\big|_{t_n^4} = \text{real\_time}\big|_{t_n^3} + d_{j,i} + w_{i,n}$$



$$\Rightarrow \begin{cases} \dfrac{1}{\alpha_j}[c_j(t_n^2) - \theta_j] = \dfrac{1}{\alpha_i}[c_i(t_n^1) - \theta_i] + d + w_{j,n} \\ \dfrac{1}{\alpha_j}[c_j(t_n^3) - \theta_j] = \dfrac{1}{\alpha_i}[c_i(t_n^4) - \theta_i] - d - w_{i,n} \end{cases}$$

Pairwise Synchronization: Gaussian case [1]

- Synchronize node j to node i (assume node i is the reference)

$$\begin{cases} \frac{1}{\alpha_j} c_j(t_n^2) = c_i(t_n^1) + \frac{\theta_j}{\alpha_j} + d + w_{j,n} \\ \frac{1}{\alpha_j} c_j(t_n^3) = c_i(t_n^4) + \frac{\theta_j}{\alpha_j} - d - w_{i,n} \end{cases}$$

- Approach 1: MLE
- Assume N rounds of time-stamp exchange, we have

$$\begin{bmatrix} c_i(t_1^1) \\ \vdots \\ c_i(t_N^1) \\ -c_i(t_1^4) \\ \vdots \\ -c_i(t_N^4) \end{bmatrix} + d\mathbf{1} = \begin{bmatrix} c_j(t_1^2) & -1 \\ \vdots & \\ c_j(t_N^2) & -1 \\ -c_j(t_1^3) & 1 \\ \vdots & \\ -c_j(t_N^3) & 1 \end{bmatrix} \begin{bmatrix} 1/\alpha_j \\ \theta_j/\alpha_j \end{bmatrix} - \begin{bmatrix} w_{j,1} \\ \vdots \\ w_{j,N} \\ w_{i,1} \\ \vdots \\ w_{i,N} \end{bmatrix} \quad \Rightarrow \quad \mathbf{t}_i + d\mathbf{1} = \mathbf{T}_j\boldsymbol{\theta} - \mathbf{z}$$

- Log-likelihood function:

$$\ln f(\mathbf{t}_i, \mathbf{T}_j \mid \boldsymbol{\theta}, d) = \ln \frac{N}{2\pi\sigma^2} - \frac{\left\| \mathbf{t}_i + d\mathbf{1} - \mathbf{T}_j\boldsymbol{\theta} \right\|^2}{2\sigma^2}$$

- $\boldsymbol{\theta}$ is linear and can be easily estimated:

$$\hat{\boldsymbol{\theta}}(d) = (\mathbf{T}_j^H \mathbf{T}_j)^{-1} \mathbf{T}_j^H (\mathbf{t}_i + d\mathbf{1})$$

- Put it back to the log-likelihood function, d can be obtained by maximizing

$$\Lambda(d) = \| \underbrace{(\mathbf{I}_{2N} - \mathbf{T}_j (\mathbf{T}_j^H \mathbf{T}_j)^{-1} \mathbf{T}_j^H)}_{=\mathbf{P}} (\mathbf{t}_i + d\mathbf{1}) \|^2$$

- Differentiating this function w.r.t. d and set it to zero:

$$\hat{d} = -\frac{1}{2} \cdot \frac{\mathbf{1}^H \mathbf{P}\mathbf{t}_i + \mathbf{t}_i^H \mathbf{P}\mathbf{1}}{\mathbf{1}^H \mathbf{P}\mathbf{1}}$$

- Finally, the clock parameters are recovered from

$$\hat{\alpha}_j = 1/[\hat{\boldsymbol{\theta}}(\hat{d})]_1, \qquad \hat{\theta}_j = [\hat{\boldsymbol{\theta}}(\hat{d})]_2 / [\hat{\boldsymbol{\theta}}(\hat{d})]_1$$

- Approach 2: Low complexity estimator
- d appears in both system model equations, so we can eliminate it by adding two equations

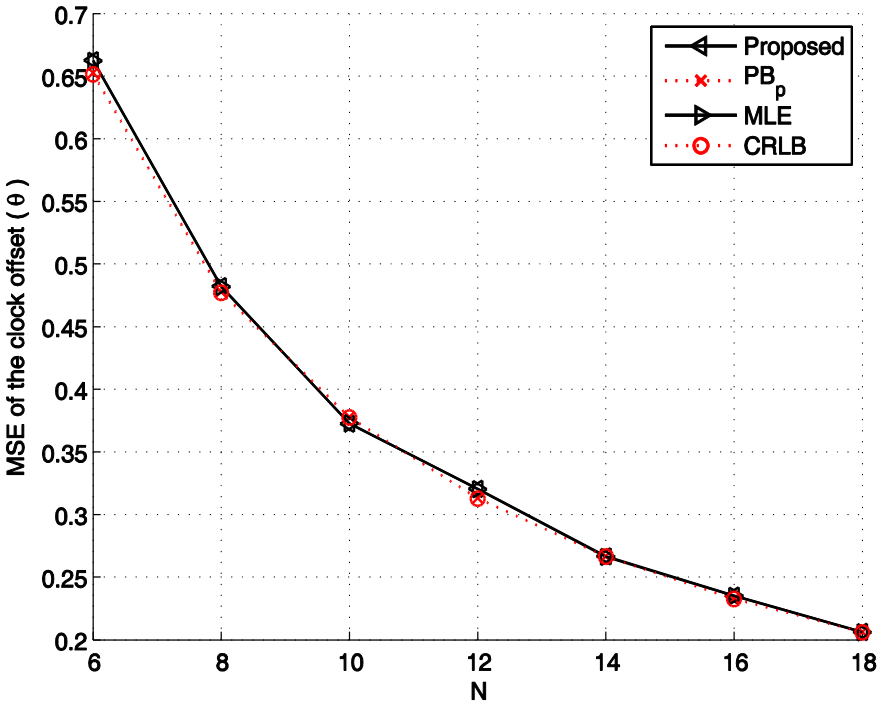
$$\frac{1}{\alpha_j}[c_j(t_n^2)+c_j(t_n^3)]=[c_i(t_n^1)+c_i(t_n^4)]+2\frac{\theta_j}{\alpha_j}+w_{j,n}-w_{i,n}$$

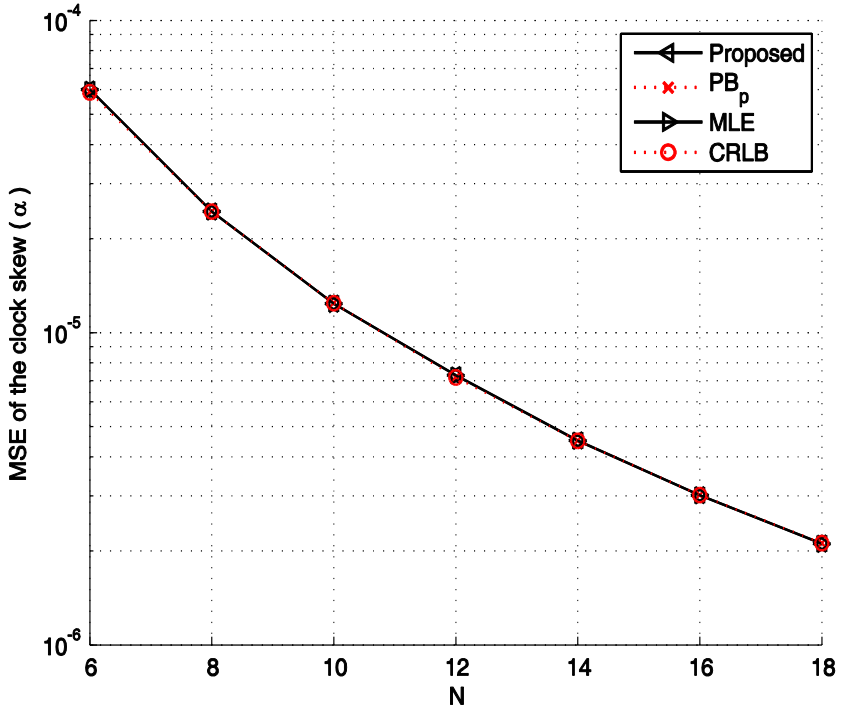
- Putting the N rounds of message into a vector form:

$$\begin{bmatrix} c_i(t_1^1)+c_i(t_1^4) \\ \vdots \\ c_i(t_N^1)+c_i(t_N^4) \end{bmatrix}=\begin{bmatrix} c_j(t_1^2)+c_j(t_1^3) & -2 \\ \vdots & \\ c_j(t_N^2)+c_j(t_N^3) & -2 \end{bmatrix}\begin{bmatrix} 1/\alpha_j \\ \theta_j/\alpha_j \end{bmatrix}+\begin{bmatrix} w_{i,1}-w_{j,1} \\ \vdots \\ w_{i,N}-w_{j,N} \end{bmatrix} \quad \Rightarrow \quad \mathbf{t}_i^{'}=\mathbf{T}_j^{'}\boldsymbol{\theta}+\mathbf{z}$$

- The MLE for this equation is $\hat{\boldsymbol{\theta}}=(\mathbf{T}_j^{'H}\mathbf{T}_j^{'})^{-1}\mathbf{T}_j^{'H}\mathbf{t}_i^{'}$
- This estimator is of lower complexity since there is no need to compute d
- But since we are not estimating the unknowns from the original equations, there may be some loss in performance

Comparison





- We have shown theoretically that the relative loss of the performance bound of the low complexity estimator w.r.t. to CRB is less than 1%

Pairwise synchronization under exponential delays [2][3]

- Approach 1: MLE
- Revisit the message exchange equations:

$$\frac{1}{\alpha_j} c_j(t_n^2) - c_i(t_n^1) - \frac{\theta_j}{\alpha_j} - d = w_{j,n}, \qquad -\frac{1}{\alpha_j} c_j(t_n^3) + c_i(t_n^4) + \frac{\theta_j}{\alpha_j} - d = w_{i,n}$$

- If $w_{j,n}$ and $w_{i,n}$ are i.i.d. exponential R.V.s, the likelihood function is (with $\beta_j=1/\alpha_j$, $\gamma_j=\theta_j/\alpha_j$)

$$f(\{c_i(t_n^1), c_j(t_n^2), c_j(t_n^3), c_i(t_n^4)\}_{n=1}^N \mid \lambda, \beta_j, \gamma_j, d) =$$

$$\lambda^{2N} \exp\left\{-\lambda \sum_{n=1}^{N} [(c_j(t_n^2) - c_j(t_n^3))\beta_j - 2d + c_i(t_n^4) - c_i(t_n^1)]\right\}$$

$$\times \prod_{n=1}^{N} \mathbb{I}[\beta_j c_j(t_n^2) - c_i(t_n^1) - \gamma_j - d \geq 0] \times \prod_{n=1}^{N} \mathbb{I}[-\beta_j c_j(t_n^3) + c_i(t_n^4) + \gamma_j - d \geq 0] \times \mathbb{I}[d \geq 0]$$

- Closed-form estimate of λ can be obtained by differentiating the above equation w.r.t. λ and set it to zero
- Putting $\hat{\lambda}$ back into the likelihood function, we can show that the MLE that maximizes the profile likelihood function is

$$[\beta_j^*, \gamma_j^*, d^*] = \arg\max_{\beta_j, \gamma_j, d} \sum_{n=1}^{N} [(c_j(t_n^2) - c_j(t_n^3))\beta_j + 2d]$$

$$\text{subject to} \begin{cases} \{\beta_j c_j(t_n^2) - c_i(t_n^1) - \gamma_j - d \geq 0\}_{n=1}^{N}; d \geq 0 \\ \{-\beta_j c_j(t_n^3) + c_i(t_n^4) + \gamma_j - d \geq 0\}_{n=1}^{N}; d \geq 0 \end{cases}$$

- This is a linear programming problem, and can be solved using existing solver, but the worst case complexity is at least $\mathcal{O}(N^3)$
- We have also proposed a low complexity algorithm for solving this problem, and the worst case only takes $\mathcal{O}(N)$

- Approach 2: Iterative weighted median
- Add the two message exchange equations:

$$\beta_j[\underbrace{c_j(t_n^2)+c_j(t_n^3)}_{=T_{r,n}}]-[\underbrace{c_i(t_n^1)+c_i(t_n^4)}_{=T_{s,n}}]-2\gamma_j = w_{j,n}-w_{i,n}$$

- $w_{j,n}$ - $w_{i,n}$ becomes Laplacian R.V. with location parameter 0 and scale parameter $1/\lambda$
- The log-likelihood function is

$$\ln f(\{T_{s,n},T_{r,n}\}_{n=1}^{N} \mid \beta_j,\gamma_j) = N\ln\frac{\lambda}{2}-\lambda\sum_{n=1}^{N}|T_{s,n}-\beta_j T_{r,n}+2\gamma_j|$$

- An estimate of β_j and γ_j can be obtained by minimizing the second term

$$\min_{\beta_j,\gamma_j}\sum_{n=1}^{N}|T_{s,n}-\beta_j T_{r,n}+2\gamma_j|$$

- Now, consider two sub-problems
  - When β_j is fixed, the problem is

$$\min_{\gamma_j} 2\sum_{n=1}^{N} | 0.5(\beta_j T_{r,n} - T_{s,n}) - \gamma_j |$$

The solution is the median value of the sequence $\left\{0.5(\beta_j T_{r,n} - T_{s,n})\right\}_{n=1}^{N}$

  - When γ_j is fixed, the problem is

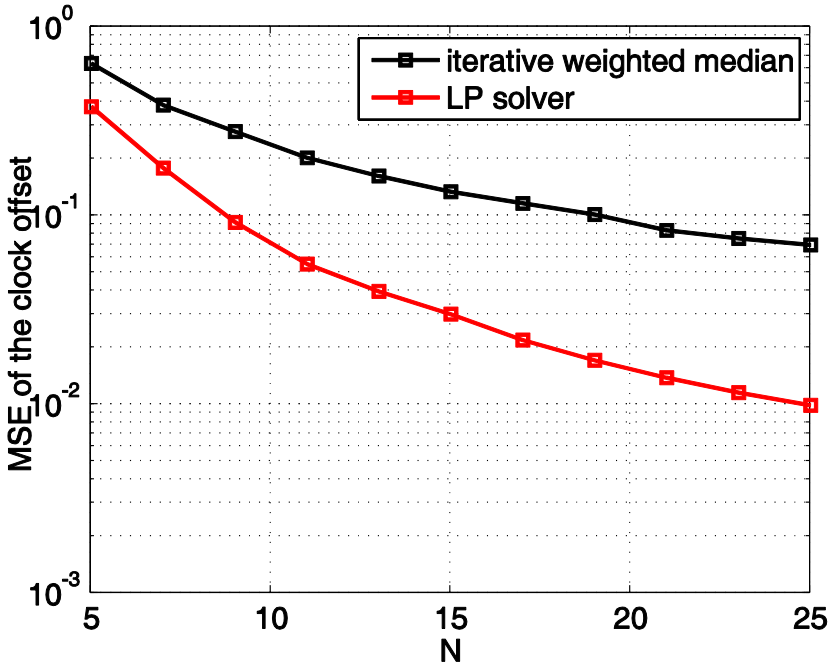
$$\min_{\beta_j} \sum_{n=1}^{N} T_{r,n} | (T_{s,n} / T_{r,n} + 2\gamma_j) - \beta_j |$$

This is a weighted median problem for the data set $\left\{T_{r,n}, (T_{s,n} / T_{r,n} + 2\gamma_j)\right\}_{n=1}^{N}$

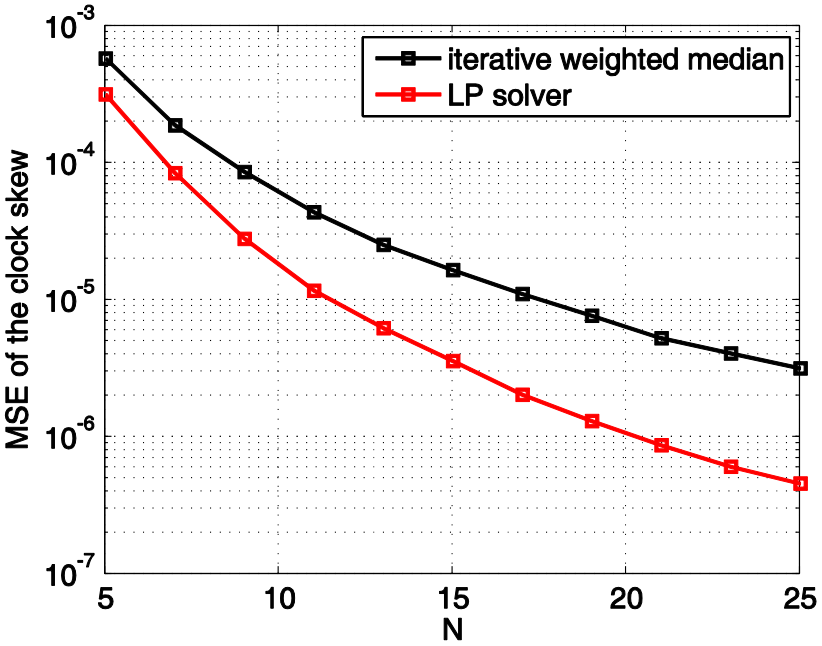
Simple procedure exists to compute this

- Two steps are iteratively updated
- Since the objective function is convex, it will converge to the global optimal solution

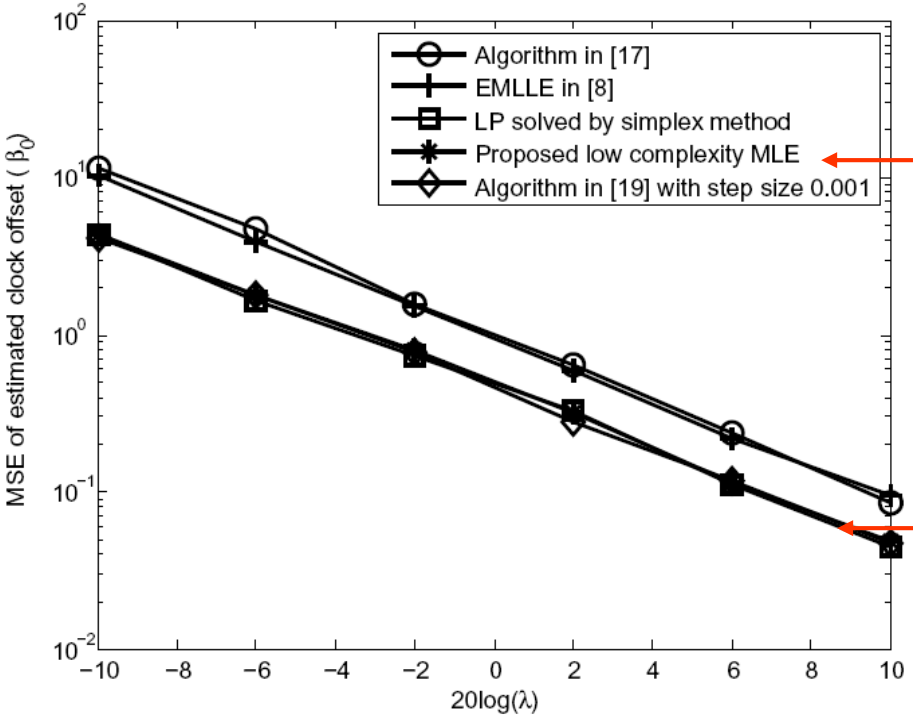
Comparison



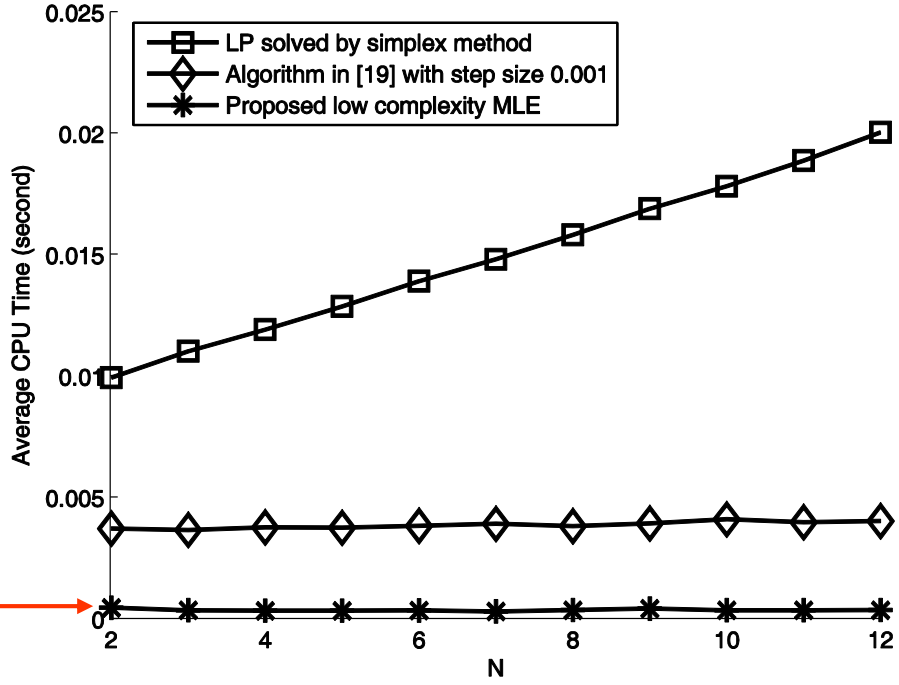
Iterative weighted median has a significant loss w.r.t. optimal solution



- The main reason is that iterative weighted median method adds the two message exchange equations together before estimation
- This is in contrast to Gaussian setting where this operation does not lead to significant loss



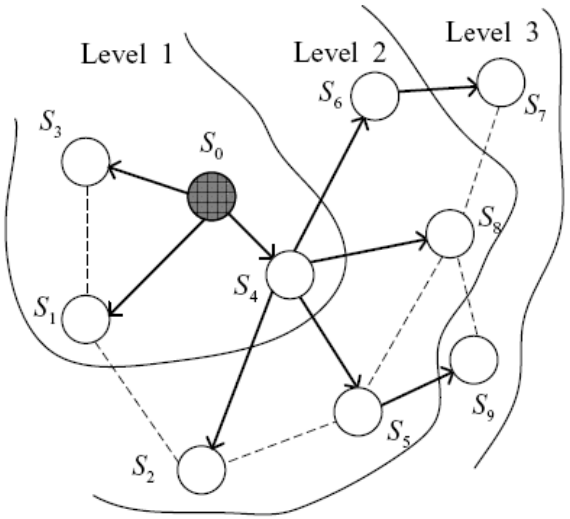
Proposed low-complexity algorithm has the same performance as LP solver



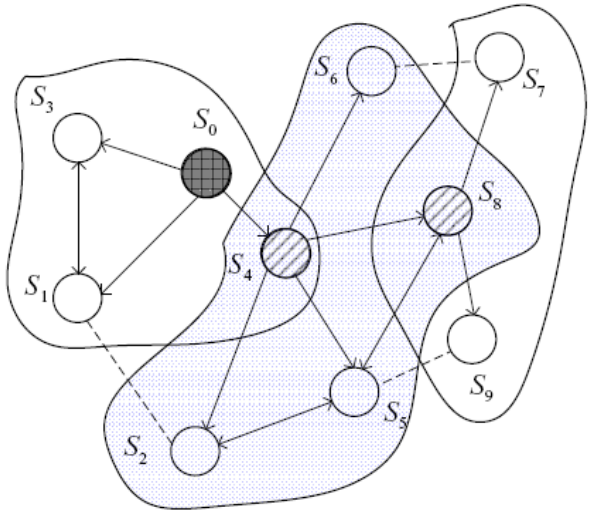
Proposed low-complexity algorithm has the lowest complexity

Network-wide synchronization

- How to extend pairwise algorithm to work for network-wide synchronization?



Tree based

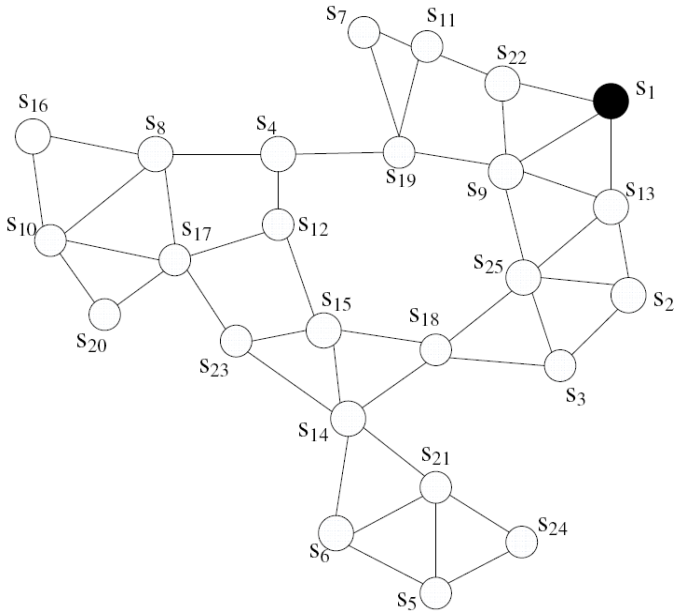


Clustered based

- Need overhead to build and maintain tree or cluster structure
- Error accumulation is quick as number of layers increases
- Vulnerable if gateway node dies

Fully Distributed Algorithms

- Approach 1: Coordinate Descent [4]
- Add the two-way message exchange equations to first eliminate d: $\frac{1}{\alpha_j}[c_j(t_n^2)+c_j(t_n^3)-2\theta_j]-\frac{1}{\alpha_i}[c_i(t_n^1)+c_i(t_n^4)-2\theta_i]=w_{j,n}-w_{i,n}$
- Assume each node perform N times two-way message exchanges with each of its direct neighbors



$$\mathrm{NLL}(\{\alpha_i,\theta_i\}_{i=2}^{M},)\propto\sum_{n=1}^{N}\sum_{i=1}^{M}\sum_{j\in\aleph(i)}\left(\frac{1}{\alpha_j}[c_j(t_n^2)+c_j(t_n^3)-2\theta_j]-\frac{1}{\alpha_i}[c_i(t_n^1)+c_i(t_n^4)-2\theta_i]\right)^2$$

But this is non-convex w.r.t. unknowns

- With transformation $\beta_j = 1/\alpha_j$ and $\gamma_j = \theta_j / \alpha_j$

$$\mathrm{NLL}(\{\beta_i, \gamma_i\}_{i=2}^{M},) \propto \sum_{n=1}^{N}\sum_{i=1}^{M}\sum_{j\in\aleph(i)}\left(\beta_j[\underbrace{c_j(t_n^2)+c_j(t_n^3)}_{\triangleq T_{r,n}^{\{i,j\}}}]-2\gamma_j-\beta_i[\underbrace{c_i(t_n^1)+c_i(t_n^4)}_{\triangleq T_{s,n}^{\{i,j\}}}]-2\gamma_i\right)^2$$

- This is convex w.r.t. β_i and γ_i, and we can alternatively minimize this NLL
- Differentiating NLL w.r.t. β_i and set it to zero (also w.r.t γ_i), we get two coupled iterative equations

$$\hat{\beta}_i^{(m+1)} = \frac{\sum_{n=1}^{N}\sum_{j\in\aleph(i)}[2\hat{\gamma}_i^{(m)}-2\hat{\gamma}_j^{(m)}][T_{s,n}^{\{i,j\}}+T_{r,n}^{\{j,i\}}]+\hat{\beta}_j^{(m)}[T_{r,n}^{\{i,j\}}\cdot T_{s,n}^{\{i,j\}}+T_{s,n}^{\{j,i\}}\cdot T_{r,n}^{\{j,i\}}]}{\sum_{n=1}^{N}\sum_{j\in\aleph(i)}[(T_{s,n}^{\{i,j\}})^2+(T_{r,n}^{\{j,i\}})^2]}$$

$$\hat{\gamma}_i^{(m+1)} = \frac{1}{4N|\aleph(i)|}\left\{\sum_{n=1}^{N}\sum_{j\in\aleph(i)}4\hat{\gamma}_j^{(m)}+\hat{\beta}_i^{(m)}[T_{s,n}^{\{i,j\}}+T_{r,n}^{\{j,i\}}]-\hat{\beta}_j^{(m)}[T_{r,n}^{\{i,j\}}+T_{s,n}^{\{j,i\}}]\right\}$$

- Approach 2: Belief Propagation [5]
- The two-way time-stamp message exchange equation (between node i and j) can be put into matrix form

$$\frac{1}{\alpha_j}[c_j(t_n^2)+c_j(t_n^3)-2\theta_j]-\frac{1}{\alpha_i}[c_i(t_n^1)+c_i(t_n^4)-2\theta_i]=w_{j,n}-w_{i,n}$$

$$\Rightarrow \begin{bmatrix} c_j(t_1^2)+c_j(t_1^3) & -2 \\ \vdots & \\ c_j(t_N^2)+c_j(t_N^3) & -2 \end{bmatrix} \begin{bmatrix} 1/\alpha_j \\ \theta_j/\alpha_j \end{bmatrix} - \begin{bmatrix} c_i(t_1^1)+c_i(t_1^4) & -2 \\ \vdots & \\ c_i(t_N^1)+c_i(t_N^4) & -2 \end{bmatrix} \begin{bmatrix} 1/\alpha_i \\ \theta_i/\alpha_i \end{bmatrix} = \begin{bmatrix} w_{j,1}-w_{i,1} \\ \vdots \\ w_{j,N}-w_{i,N} \end{bmatrix}$$

$$\Rightarrow \quad \mathbf{A}_{j,i}\boldsymbol{\beta}_j - \mathbf{A}_{i,j}\boldsymbol{\beta}_i = \mathbf{z}_{j,i}$$

- Marginalized posterior distribution at node i:

$$g(\boldsymbol{\beta}_i) \propto \int \cdots \int \prod_{i=1}^{M} p(\boldsymbol{\beta}_i) \prod_{\{i,j\}\in E}^{M} p(\mathbf{A}_{i,j}, \mathbf{A}_{j,i} \mid \boldsymbol{\beta}_i, \boldsymbol{\beta}_j) d\boldsymbol{\beta}_1 \ldots d\boldsymbol{\beta}_{i-1} d\boldsymbol{\beta}_{i+1} \ldots d\boldsymbol{\beta}_M$$

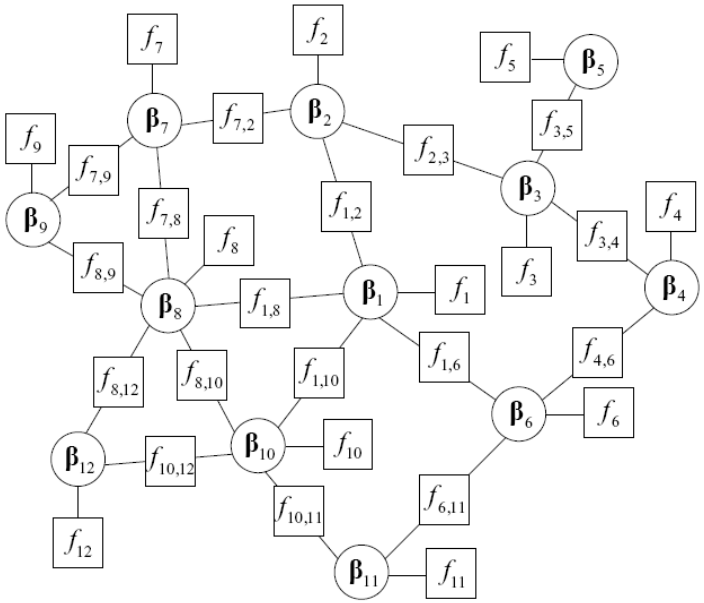
- Computational demanding, needs centralized processing

- Express the joint posterior distribution using factor graph
  - Factor node: local likelihood function
  - or prior distribution

$$f_{i,j} = p(\mathbf{A}_{i,j}, \mathbf{A}_{j,i} \mid \boldsymbol{\beta}_i, \boldsymbol{\beta}_j)$$
$$= \mathcal{N}(\mathbf{A}_{j,i}\boldsymbol{\beta}_j \mid \mathbf{A}_{i,j}\boldsymbol{\beta}_i, \sigma_{i,j}^2 \mathbf{I}_N)$$

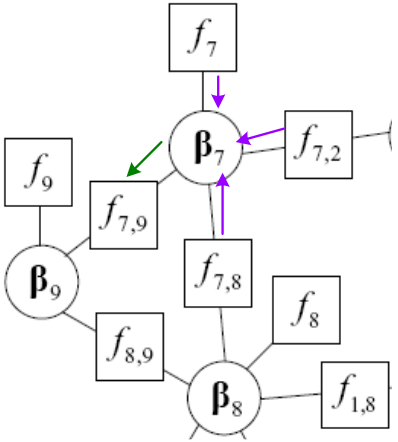
$$f_i = p(\boldsymbol{\beta}_i)$$

  - Variable node: β_i



- Marginal distribution at each node can be obtained by message passing on factor graph:
  - Message from variable node to factor node

$$m_{j \to f_{i,j}}^{(l)}(\boldsymbol{\beta}_j) = \prod_{f \in B(\beta_j) \setminus f_{i,j}} m_{f \to j}^{(l)}(\boldsymbol{\beta}_j)$$



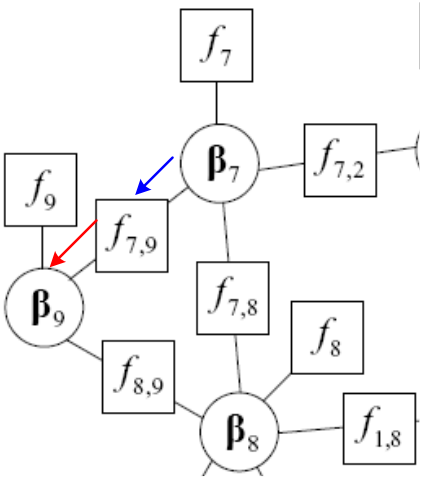
– Message from factor node to variable node:

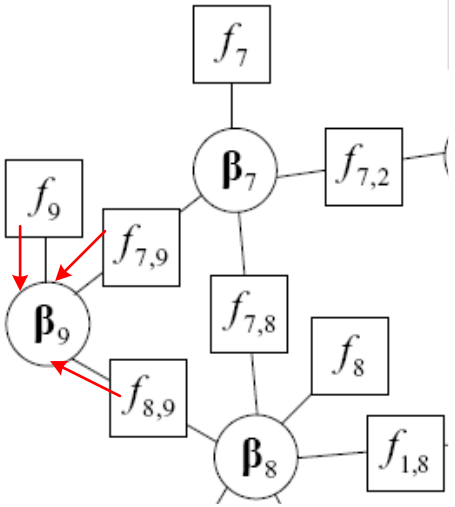
$$m^{(l)}_{f_{i,j}\to i}(\boldsymbol{\beta}_i) = \int m^{(l-1)}_{j\to f_{i,j}}(\boldsymbol{\beta}_j)\, f_{i,j}\, d\boldsymbol{\beta}_j$$

– In each iteration, messages are updated in parallel with the information from direct neighbouring nodes

– Each node computes the belief locally:

$$b^{(l)}(\boldsymbol{\beta}_i) = \prod_{f\in B(\boldsymbol{\beta}_i)} m^{(l)}_{f\to i}(\boldsymbol{\beta}_i)$$





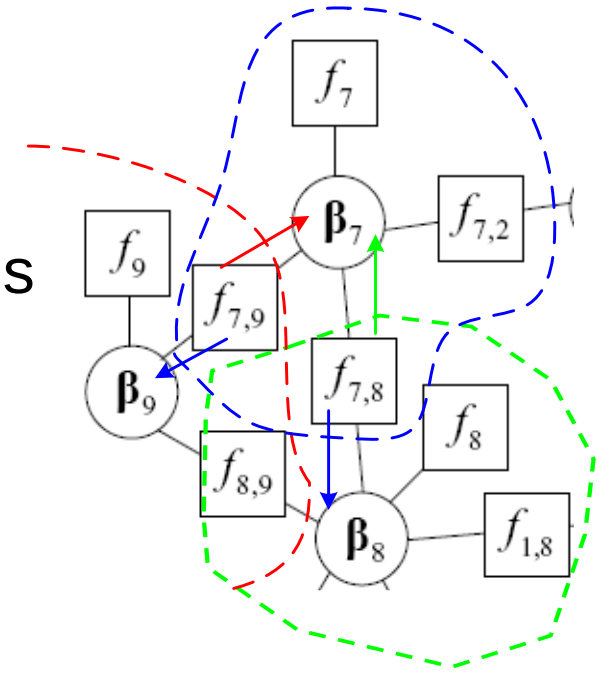
- As the likelihood function is Gaussian, the messages involved in this algorithm keep the Gaussian form

$$m^{(l)}_{i\to f_{i,j}}(\boldsymbol{\beta}_i) \sim \mathcal{N}(\boldsymbol{\beta}_i \mid \mathbf{v}^{(l)}_{i\to f_{i,j}}, \mathbf{C}^{(l)}_{i\to f_{i,j}})$$

$$m^{(l)}_{f_{j,i}\to i}(\boldsymbol{\beta}_i) \sim \mathcal{N}(\boldsymbol{\beta}_i \mid \mathbf{v}^{(l)}_{f_{j,i}\to i}, \mathbf{C}^{(l)}_{f_{j,i}\to i})$$

- In practice, each real node computes both kinds of messages
- Information passes between real nodes is set as **message from factor-to-variable**, and inherits the properties
  - Still Gaussian. Only mean and covariance needed to be exchanged
  - Updated in parallel by local computation with received mean and covariance messages from neighboring nodes
- The belief computed at node i, with the received messages from all direct neighbouring nodes, is still Gaussian: $b^{(l)}(\boldsymbol{\beta}_i) \sim \mathcal{N}\left(\boldsymbol{\beta}_i \,|\, \boldsymbol{\mu}_i^{(l)}, \mathbf{P}_i^{(l)}\right)$ where

$$\left[\mathbf{P}_i^{(l)}\right]^{-1} = \sum_{j \in \aleph(i)} \left[\mathbf{C}_{f_{i,j} \to i}^{(l)}\right]^{-1} \qquad \boldsymbol{\mu}_i^{(l)} = \mathbf{P}_i^{(l)} \sum_{j \in \aleph(i)} \left[\mathbf{C}_{f_{i,j} \to i}^{(l)}\right]^{-1} \mathbf{v}_{f_{i,j} \to i}^{(l)}$$

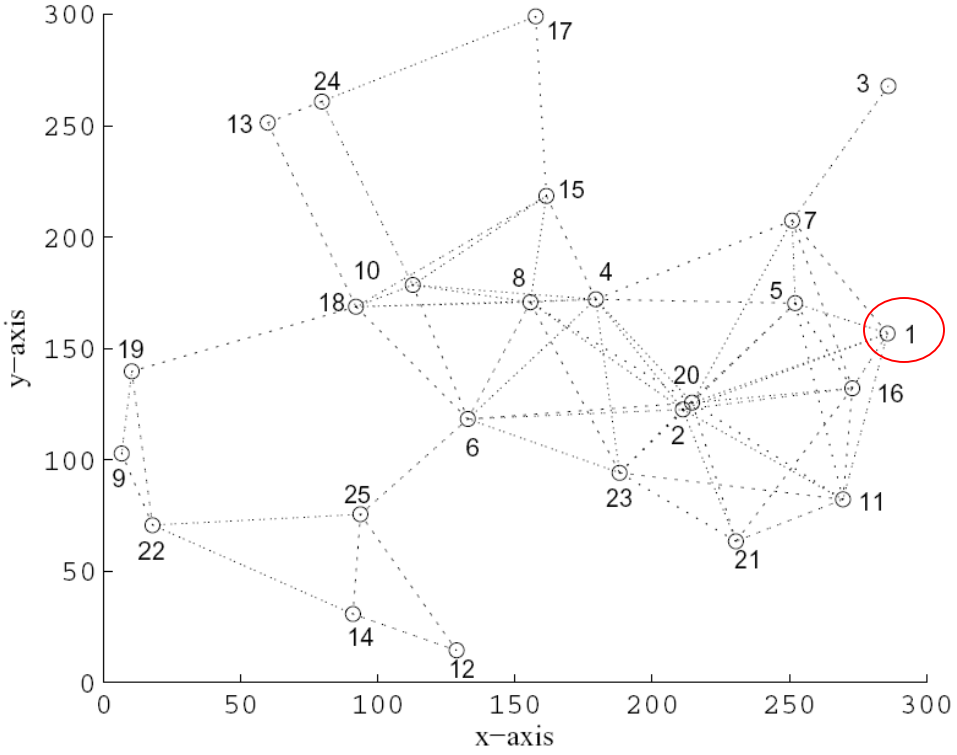


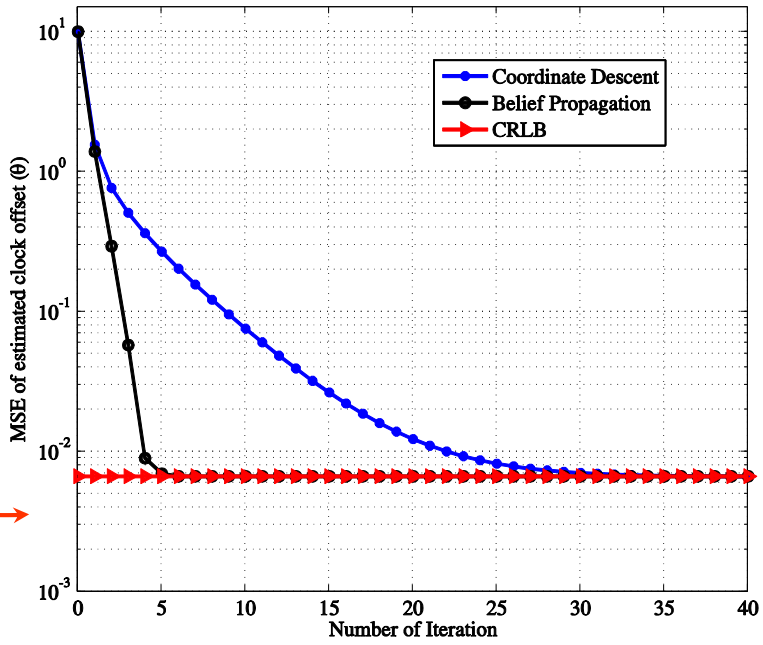
- The estimate at node i is $\hat{\boldsymbol{\beta}}_i^{(l)} = \int \boldsymbol{\beta}_i b^{(l)}(\boldsymbol{\beta}_i) d\boldsymbol{\beta}_i = \boldsymbol{\mu}_i^{(l)}$
- The θ_i and α_i can be recovered from $\hat{\boldsymbol{\beta}}_i$ after convergence
- It is generally known that if the FG contains cycles, messages can flow many times around the graph, leading to the possibility of divergence of BP algorithm

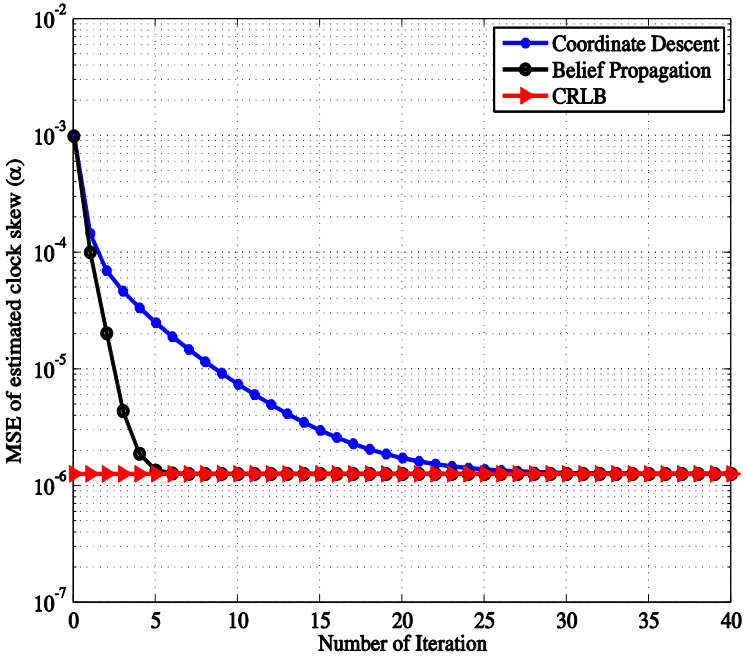
- Two properties:
- BP in this application converges regardless of network topology, even under asynchronous message update
- The converged solution can also be proved to be equal to the centralized ML solution

Comparison

- Simulation setting: 25 nodes, $d_{ij} \in$[8,12], $\theta_i \in$[-5.5,5.5], $\alpha_i \in$[-0.955,1.055], $\sigma_i^2 = 0.1$, 1000 topologies, initialization=[1, 0], N=10

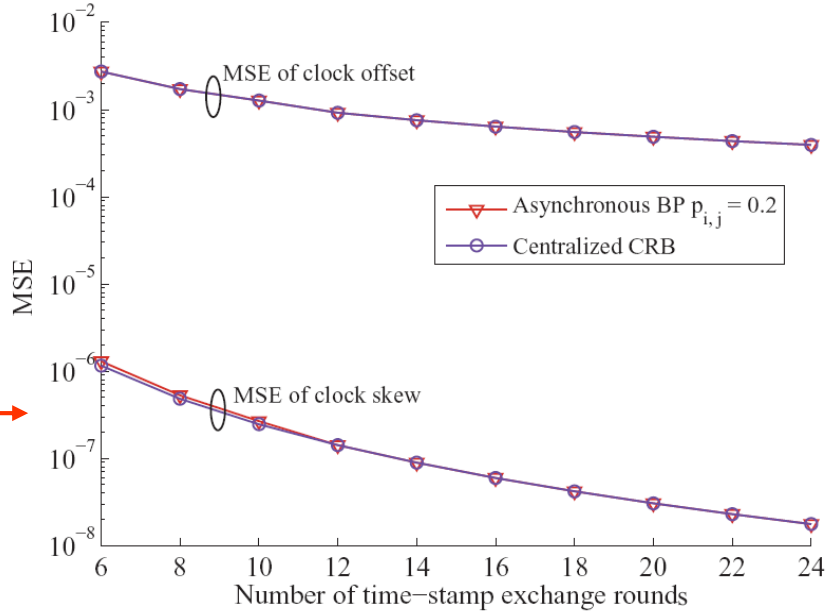






BP converges much faster than CD as second order information is included in the messages

BP algorithm under asynchronous message exchange



Distributed tracking with DKF [4]

- Clock parameters may stay constant within a short period of time
- But it will change over time
- We can either redo synchronization (throwing away previous estimates), or we can do tracking
- If the change is slow, tracking is preferred
- Re-representation of clock model:

$$c_i(t) = \alpha_i t + \theta_i^0 + \sqrt{p_i} B(t)$$

This term is due to phase noise

$$= \theta_i^0 + \int_0^t \beta_i(\tau) d\tau$$

$$\beta_i(t) = \alpha_i + \sqrt{p_i} B'(t)$$

- After sampling:

$$c_i(l) = l\tau_0 + \underbrace{\underbrace{\sum_{m=0}^{l-1} [\beta_i(m) - 1]\tau_0 + \theta_i^0}_{\triangleq \vartheta_i(l-1)} + [\beta_i(l) - 1]\tau_0}_{\vartheta_i(l)}$$

- Writing the accumulated skew and offset in recursive forms:

$$\beta_i(l) = \beta_i(l-1) + \underbrace{\sqrt{p_i}\,[B'(l) - B'(l-1)]}_{\triangleq u_i(l)}$$

Gaussian with zero mean and variance $2p_i$

$$\vartheta_i(l) = \vartheta_i(l-1) + [\beta_i(l) - 1]\tau_0$$

- Clock parameter evolution model:

$$\underbrace{\begin{bmatrix}\beta_i(l)\\ \vartheta_i(l)\end{bmatrix}}_{\triangleq \mathbf{x}_i(l)} = \underbrace{\begin{bmatrix}1 & 0\\ \tau_0 & 1\end{bmatrix}}_{\triangleq \mathbf{A}_i}\begin{bmatrix}\beta_i(l-1)\\ \vartheta_i(l-1)\end{bmatrix} + \begin{bmatrix}u_i(l)\\ \tau_0 u_i(l)\end{bmatrix} + \underbrace{\begin{bmatrix}0\\ -\tau_0\end{bmatrix}}_{\triangleq \mathbf{b}_i}$$

- Measurement equations based on two-way message exchange

$$T_{r,n}^{\{i,j\}} - T_{s,n}^{\{i,j\}} = 2\vartheta_j(l) - 2\vartheta_i(l) + V_l^{\{i,j\}}$$

- Gather all measurement equations for $j \in \aleph(i)$

$$\mathbf{z}_{i,l} = \tilde{\mathbf{C}}_{i,l}\mathbf{x}_{\aleph(i)}(l) + \mathbf{v}_i(l)$$

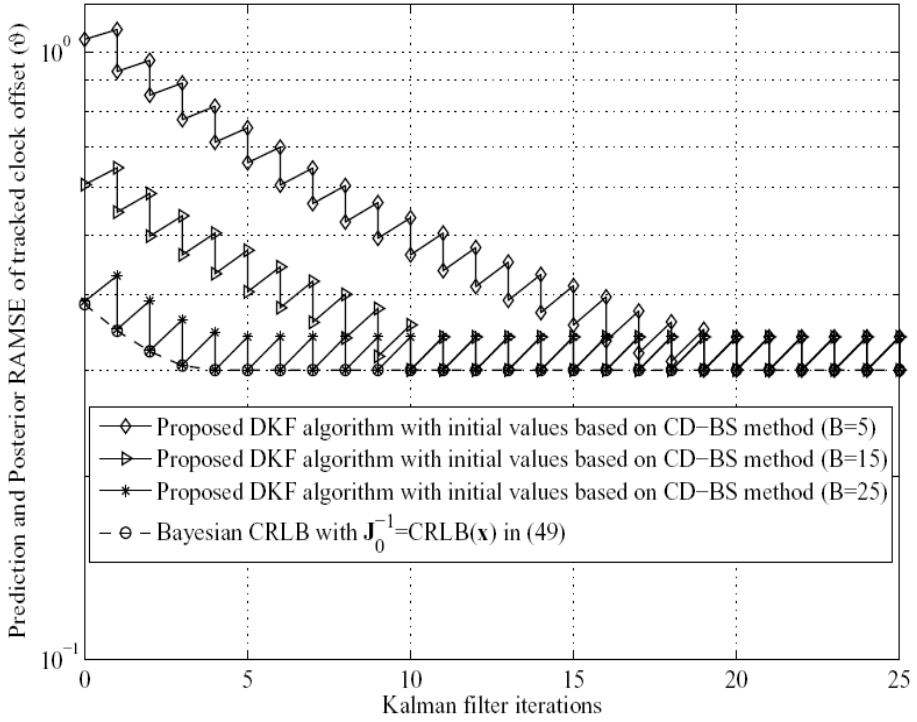
- If all information is gathered in a single place, the optimal solution is centralized KF
- KF cannot be implemented in distributed way since the Kalman gain matrix contains correlation among nodes
- Solution: Impose a block diagonal structure on the Kalman gain matrix:

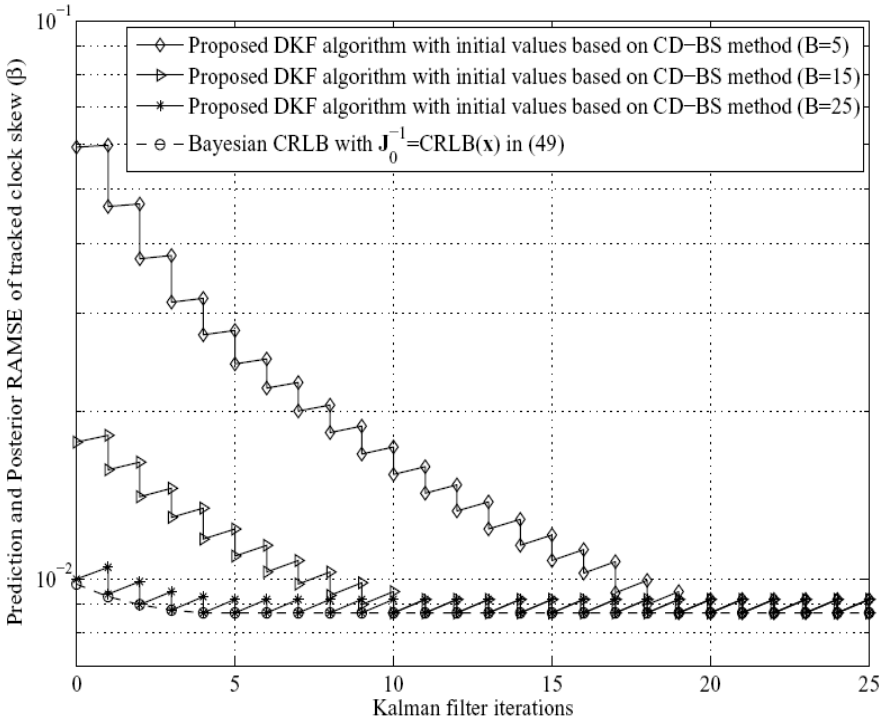
$$\begin{cases} \hat{\mathbf{x}}_i(l_k \mid l_{k-1}) = \mathbf{A}_i \hat{\mathbf{x}}_i(l_{k-1} \mid l_{k-1}) + \mathbf{b}_i \\ \hat{\mathbf{x}}_i(l_k \mid l_k) = \hat{\mathbf{x}}_i(l_k \mid l_{k-1}) + \mathbf{K}_i(l_k)(\mathbf{z}_{i,l_k} - \tilde{\mathbf{C}}_{i,l}\mathbf{x}_{\aleph(i)}(l_k \mid l_{k-1})) \end{cases}$$

$$\mathbf{K}(l_k) = \arg\min_{\mathbf{K}(l_k)} \mathrm{Tr}\mathbf{P}(l_k \mid l_k)$$

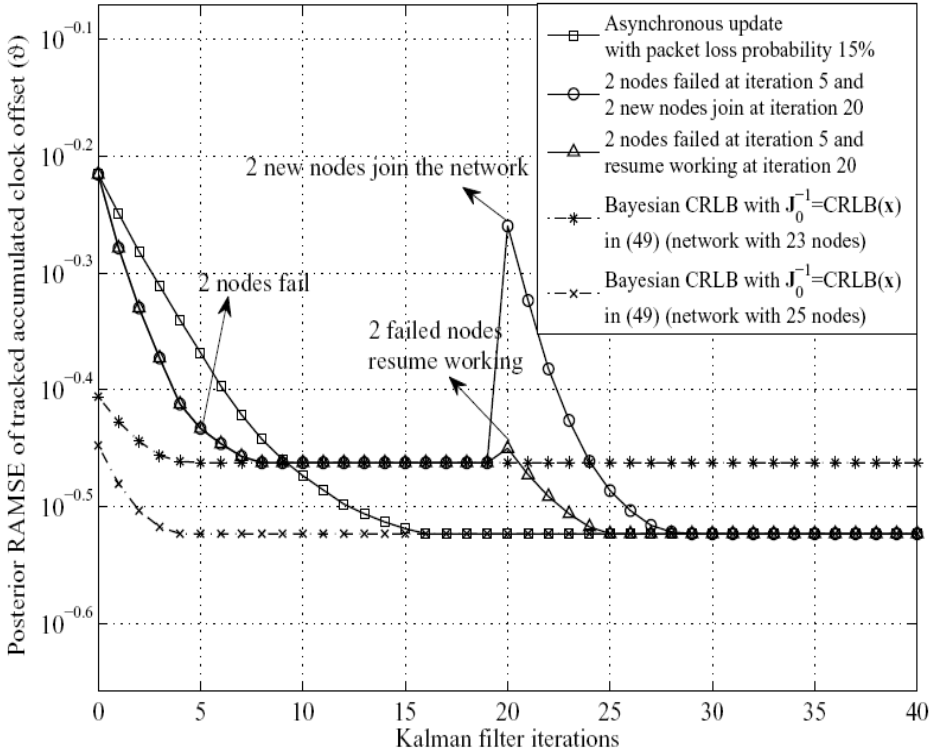
$$\text{s.t. } \mathbf{K}(l_k) = \sum_{i=2}^{M} \mathbf{U}_i^T \mathbf{K}_i(l_k)\boldsymbol{\Omega}_i$$

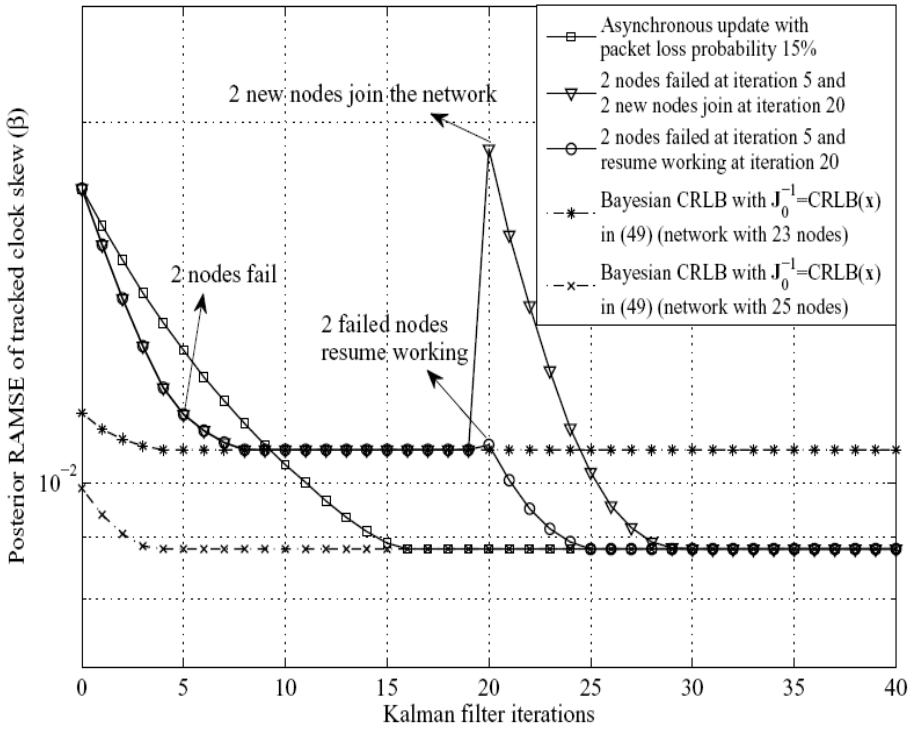
- We can solve for $\mathbf{K}_i(l_k)$ in closed form
- Each round of tracking includes time-stamp exchange and messages exchange for KF update





- Initialization:
  - $\mathbf{x}_i(0|0) = [1 \;\; 0]^T$, $\mathbf{P}(0|0) = \delta^{-1}\mathbf{I}$ but it takes a long time to converge
  - CD + bootstrap for covariance estimation (5 rounds of initial time-stamp exchange)





- Start with 25 nodes (B=15)
- If a node fails during tracking, we simply remove it from the equations
- If the node later resume working, we will use its previously stored clock parameter estimates and covariance matrix
- If a new node suddenly join in, we will use $\mathbf{x}_i(0|0)=[1\ \ 0]^T$, $\mathbf{P}_i(0|0)=\delta^{-1}\mathbf{I}$

Distributed algorithm under exponential delays [6]

- The pairwise LP problem can be easily extended to network-wide setting:

$$[\mathbf{x}^*, \mathbf{d}^*] = \arg\max_{\mathbf{x},\mathbf{d}} \sum_{i=1}^{M} \sum_{j \in \aleph(i)} \left\{ \beta_j \sum_{n=1}^{N} (c_j(t_n^2) - c_j(t_n^3)) + \beta_i \sum_{n=1}^{N} (c_i(t_n^4) - c_i(t_n^1)) - 2Nd_{ij} \right\}$$

$$\text{subject to} \begin{cases} c_j(t_n^2)\beta_j - c_i(t_n^1)\beta_i - \gamma_j + \gamma_i - d_{ij} \geq 0 \\ -c_j(t_n^3)\beta_j + c_i(t_n^4)\beta_i + \gamma_j - \gamma_i - d_{ij} \geq 0 \qquad \forall (i,j) \in E,\ n = 1, \ldots, N \\ d_{ij} \geq 0 \end{cases}$$

- This is a LP problem, but very large in size
- Centralized solution is computational expensive and has large communication overhead

- Challenge of solving this problem in a distributed way: constraints are coupled for parameters at different nodes
- By introducing slack variables $\mathbf{w}$ and auxiliary replica variables $\mathbf{z}$, we can transform the problem to

$$\arg\min_{\mathbf{x},\mathbf{d},\mathbf{w},\mathbf{z}} \sum_{i=2}^{M} \mathbf{a}_i^T \mathbf{x}_i + \sum_{i=1}^{M} \sum_{j\in\aleph(i)} (-2N) d_{ij} + \mathbb{I}(\mathbf{d} \geq \mathbf{0}) + \mathbb{I}(\mathbf{w} \geq \mathbf{0})$$

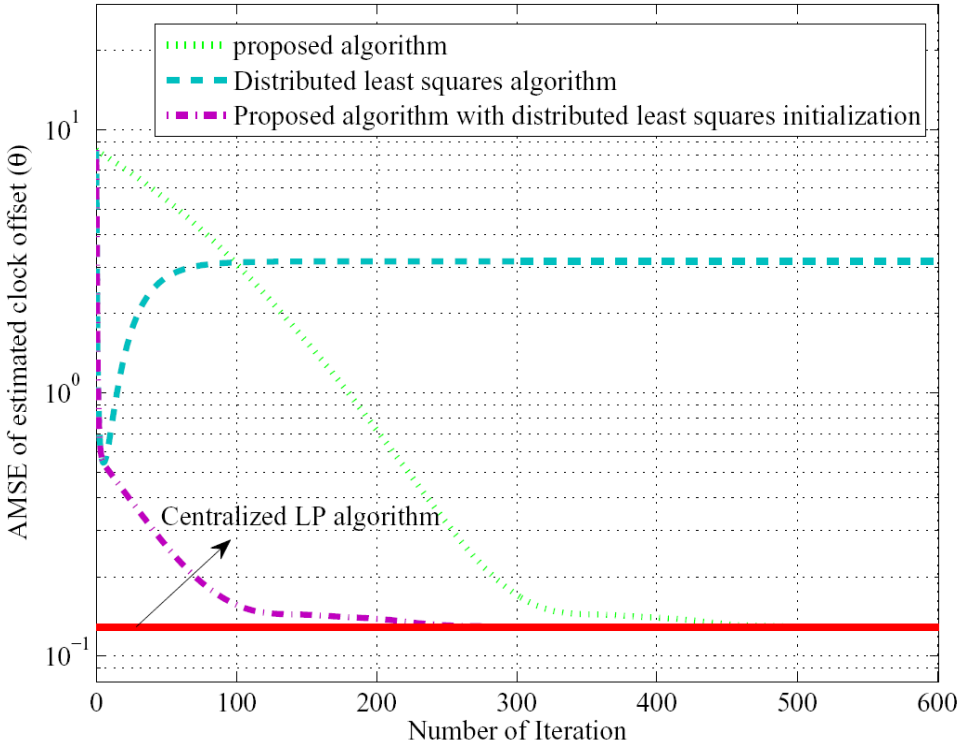
$$\text{s.t. } \mathbf{B}^{\{i,j\}}\mathbf{x}_i = \mathbf{z}_1^{\{i,j\}},\ \mathbf{E}^{\{i,j\}}\mathbf{x}_j = \mathbf{z}_2^{\{i,j\}},$$

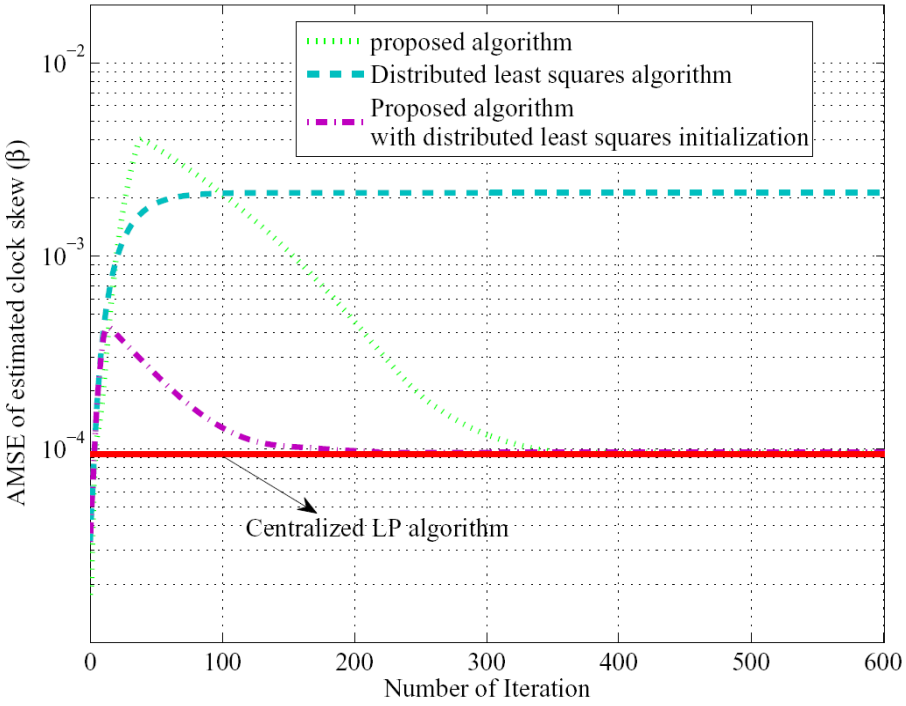
$$d_{ij}\mathbf{1}_{2N} = \mathbf{z}_3^{\{i,j\}},\ \mathbf{w}_{ij} = \mathbf{z}_4^{\{i,j\}},$$

$$\sum_{q=1}^{4} \mathbf{z}_q^{\{i,j\}} = \mathbf{0} \quad \forall (i,j) \in E$$

- This problem can be solved by ADMM (iteratively minimizing the augmented Lagrangian function w.r.t. the unknowns, $\mathbf{x}$, $\mathbf{d}$, $\mathbf{w}$, $\mathbf{z}$, and the Lagrange multipliers)

- Properties:
  - The resultant algorithm only involves local computation at each node and communications with its direct neighbors
  - Closed-form expressions are available for each update step
  - It will converge to the centralized ML solution
- Contrast to existing applications of ADMM:
  - Direct application of ADMM to the original LP would not result in distributed algorithm
  - Most existing applications that result in distributed algorithms are for Gaussian likelihood
  - Most existing works consider a single (or a small set of) common parameter





- Simulation results on a 25 nodes network
- K=5
- λ=1
- CD as initialization help to speed up convergence

Conclusions

- We discussed clock synchronization in wireless sensor networks
- We started with pariwise synchronization: Gaussian case and exponential case
- Then we discussed network-wide synchronization: CD, BP, ADMM
- We also discussed tracking using distributed KF
- Future works:
  - BP based distributed algorithm for exponential delays?
  - How about arbitrary distributed delays, asymmetric delays?

References

[1] Mei Leng and Yik-Chung Wu, ``On Clock Synchronization Algorithms for Wireless Sensor Networks under Unknown Delay," *IEEE Trans. on Vehicular Technology*, vol. 59, no.1, pp. 182-190, Jan 2010.

[2] Mei Leng and Yik-Chung Wu, ``Low Complexity Maximum Likelihood Estimators for Clock Synchronization of Wireless Sensor Nodes under Exponential Delays," *IEEE Trans. on Signal Processing*, Vol. 59, no. 10, pp. 4860-4870, Oct 2011.

[3] Mei Leng and Yik-Chung Wu, ``On joint synchronization of clock offset and skew for Wireless Sensor Networks under exponential delay," *Proceedings of the IEEE ISCAS 2010*, Paris, France, pp. 461-464, May 2010.

[4] Bin Luo and Yik-Chung Wu, ``Distributed Clock Parameters Tracking in Wireless Sensor Network," *IEEE Trans. on Wireless Communications,* Vol. 12, no. 12, pp.6464-6475, Dec 2013.

[5] Jian Du and Yik-Chung Wu, ``Distributed Clock Skew and Offset Estimation in Wireless Sensor Networks: Asynchronous Algorithm and Convergence Analysis," *IEEE Trans. on Wireless Communications,* Vol. 12, no. 11, pp. 5908-5917, Nov. 2013.

[6] Bin Luo, Lei Cheng, and Yik-Chung Wu, ``Fully-distributed Clock Synchronization in Wireless Sensor Networks Under Exponential Delays," *Signal Processing,* Vol. 125, pp. 261-273, Aug 2016. http://www.sciencedirect.com/science/article/pii/S0165168416000578

Further related readings:

- Yik-Chung Wu, Qasim M. Chaudhari and Erchin Serpedin, ``Clock Synchronization of Wireless Sensor Networks," *IEEE Signal Processing Magazine*, Vol. 28, no. 1, pp.124-138, Jan. 2011.

- Jun Zheng and Yik-Chung Wu, ``Joint Time Synchronization and Localization of an unknown node in Wireless Sensor Networks," *IEEE Trans. on Signal Processing*, Vol. 58, no. 3, pp. 1309-1320, Mar 2010.
- Mei Leng and Yik-Chung Wu, ``Distributed Clock Synchronization for Wireless Sensor Networks using Belief Propagation," *IEEE Trans. on Signal Processing*, Vol. 59, no. 11, pp. 5404-5414, Nov 2011.