Optimal sampling of multiple linear processes over a shared medium

Sebin Mathew[a], Karl H. Johansson[b], Aditya Mahajan[a]

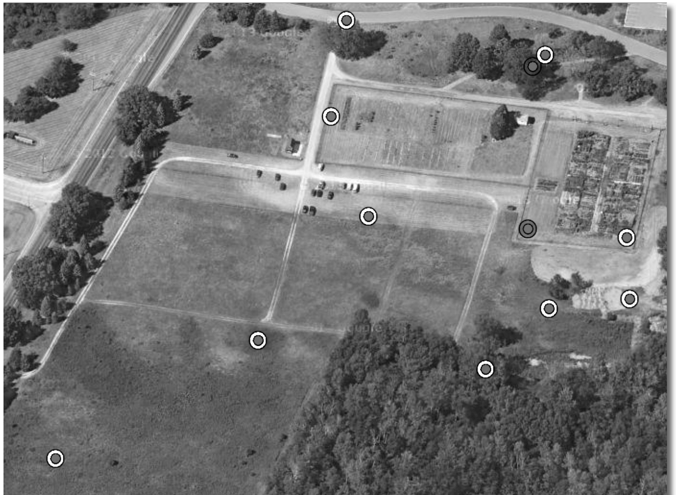
[a] McGill University

[b] KTH Royal Institute of Technology

IEEE Conference on Decision and Control
17 December, 2018

Many remote estimation applications where:

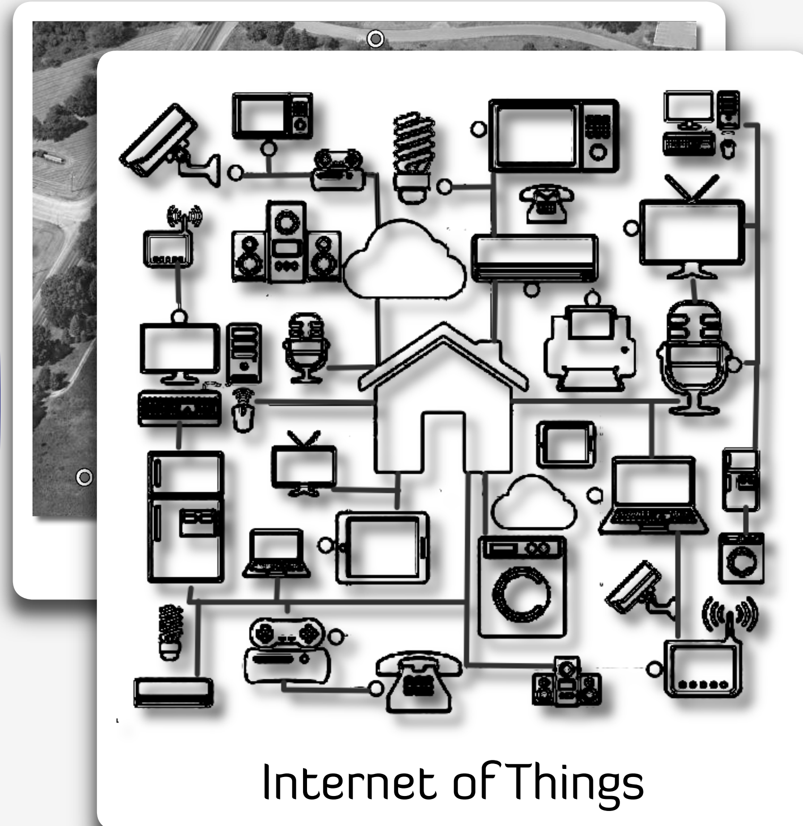
- Multiple sensors transmit over shared links
- Link capacity varies exogenously



Sensor Networks

Many remote estimation applications where:

- Multiple sensors transmit over shared links
- Link capacity varies exogenously



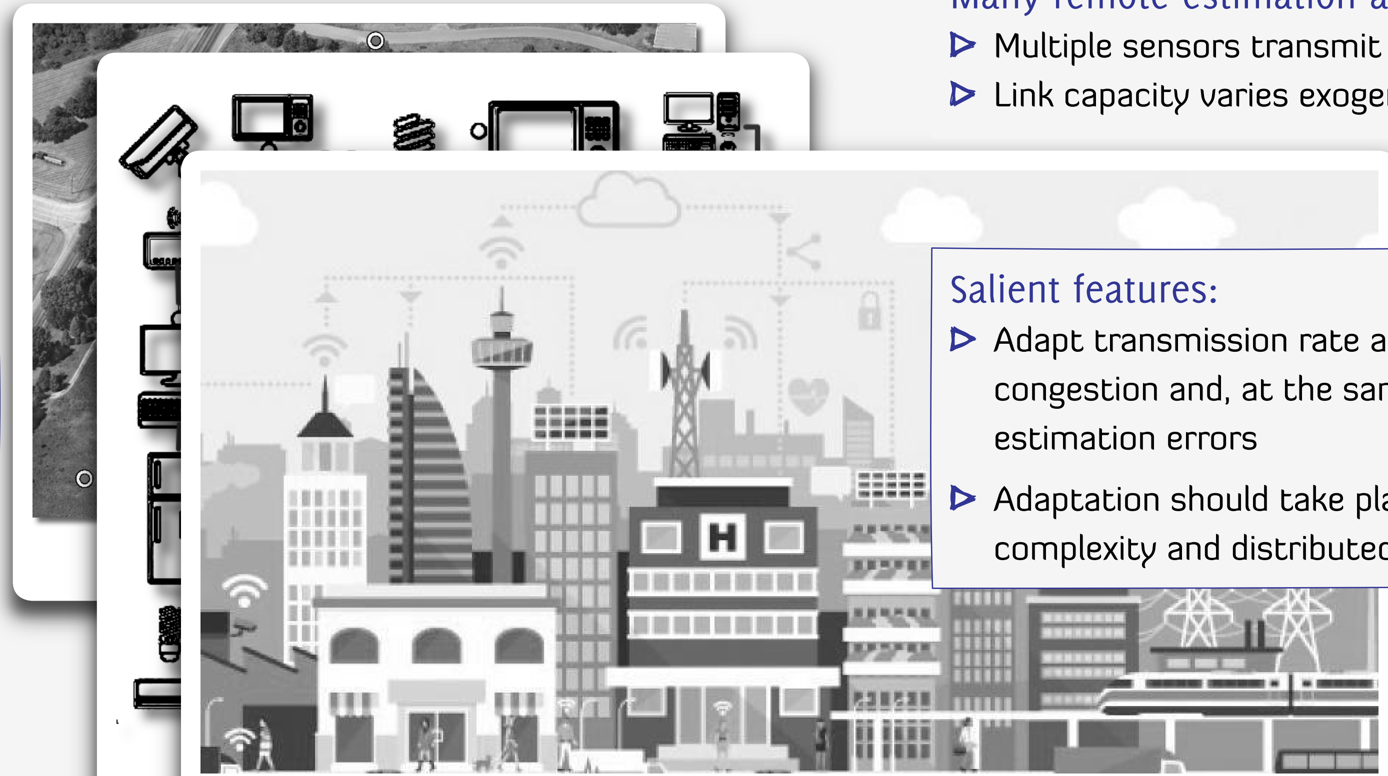
Many remote estimation applications where:

- Multiple sensors transmit over shared links
- Link capacity varies exogenously

Many remote estimation applications where:

- Multiple sensors transmit over shared links
- Link capacity varies exogenously

Smart Cities



Smart Cities

Many remote estimation applications where:

- Multiple sensors transmit over shared links
- Link capacity varies exogenously

Salient features:

- Adapt transmission rate at sensors to avoid congestion and, at the same time, minimize estimation errors
- Adaptation should take place in a low complexity and distributed manner

Many remote estimation applications where:

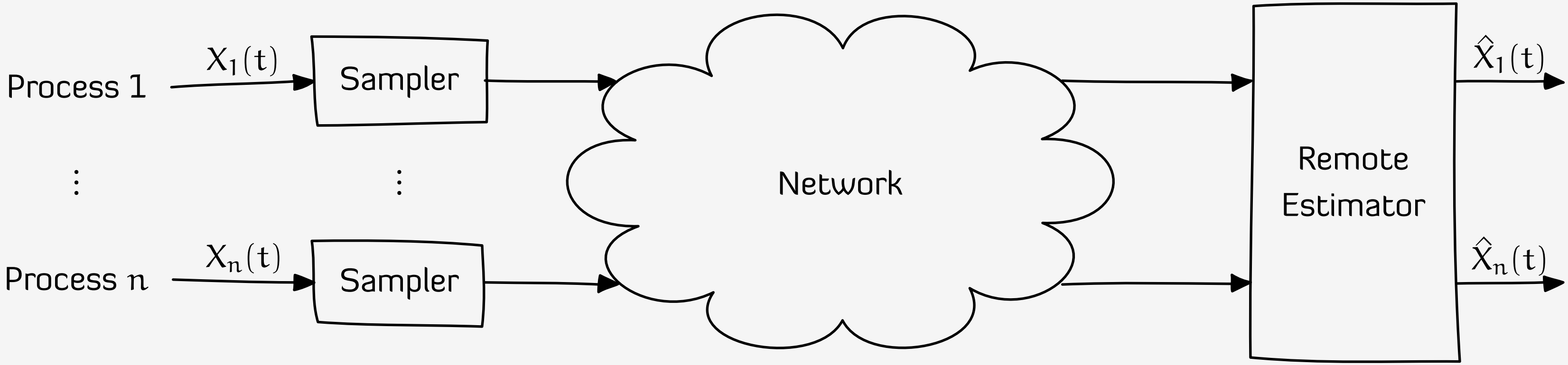
- Multiple sensors transmit over shared links
- Link capacity varies exogenously

Salient features:

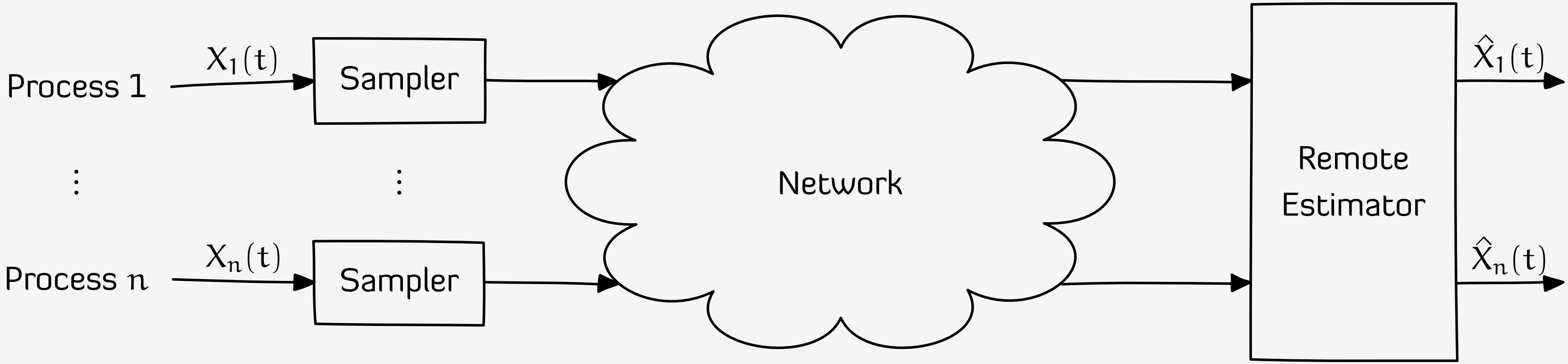
- Adapt transmission rate at sensors to avoid congestion and, at the same time, minimize estimation errors
- Adaptation should take place in a low complexity and distributed manner

Show that such questions can be answered using dual decomposition theory

System Model

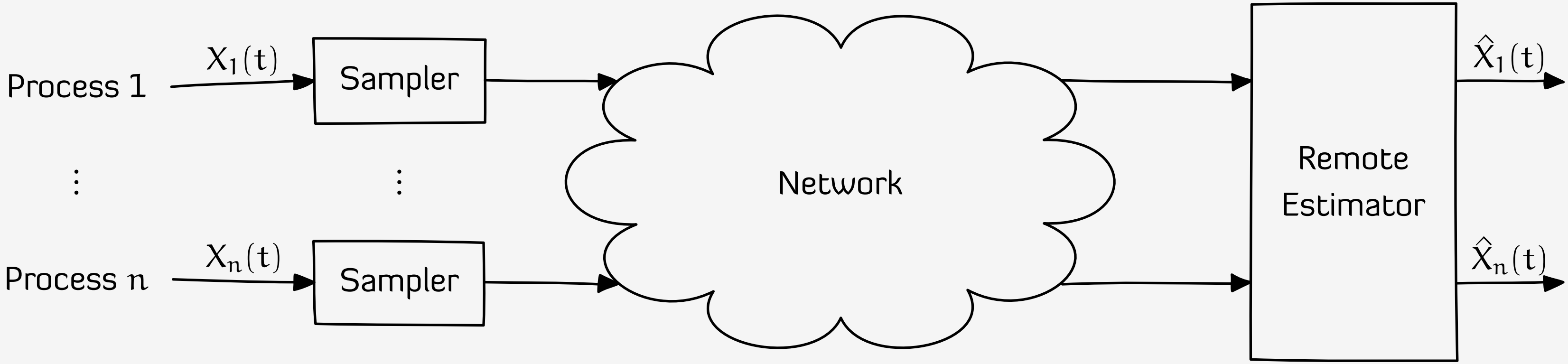


System Model



Process dynamics $\quad dX_i(t) = a_i X_i(t)dt + dW_i(t)$. $\quad \{W_i(t)\}_{t \geq 0}$ is stationary and indep across sensors.

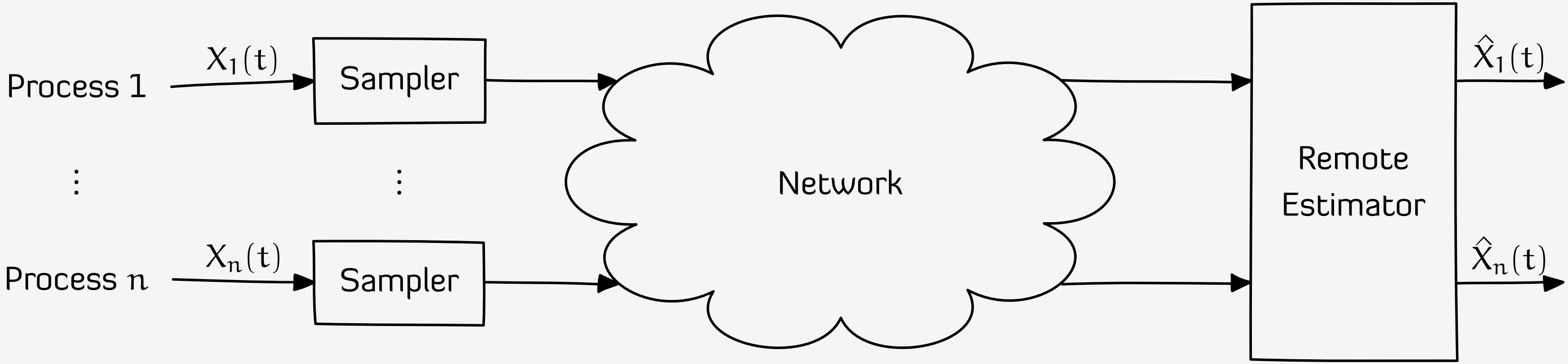
System Model



Process dynamics $dX_i(t) = a_i X_i(t)dt + dW_i(t).$ $\{W_i(t)\}_{t\geq 0}$ is stationary and indep across sensors.

Sampling process Sensor i samples process i at rate $R_i = 1/T_i$.

System Model

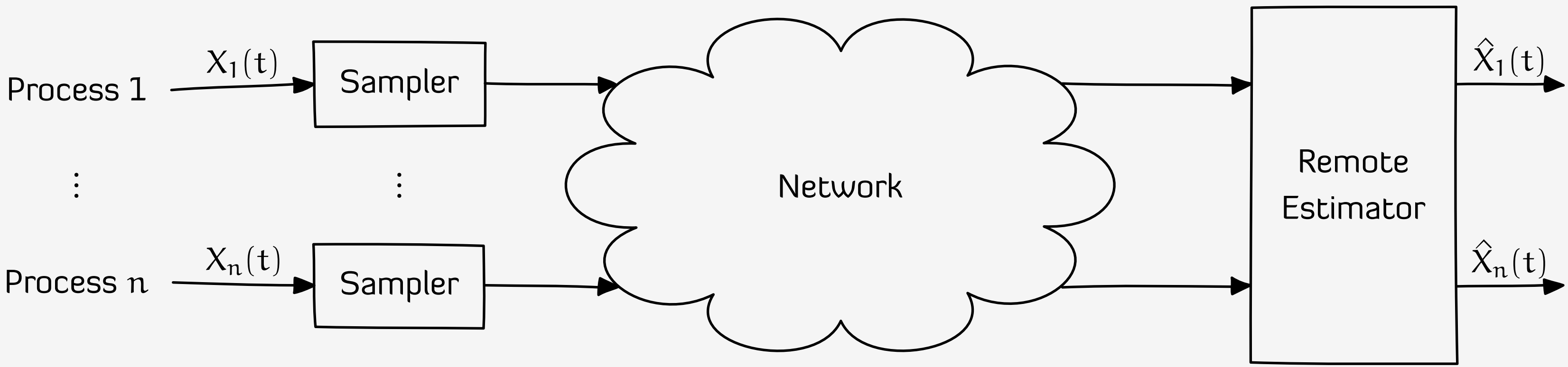


Process dynamics $dX_i(t) = a_i X_i(t)dt + dW_i(t).$ $\{W_i(t)\}_{t \geqslant 0}$ is stationary and indep across sensors.

Sampling process Sensor i samples process i at rate $R_i = 1/T_i$.

Network Rate region $\mathcal{R} = \{(R_1, \ldots, R_n) \in \mathbb{R}^n_{\geqslant 0} : \sum_{i=1}^n R_i \leqslant C\}$

System Model



Process dynamics $dX_i(t) = a_i X_i(t)dt + dW_i(t)$. $\{W_i(t)\}_{t \geqslant 0}$ is stationary and indep across sensors.

Sampling process Sensor i samples process i at rate $R_i = 1/T_i$.

Network Rate region $\mathcal{R} = \{(R_1, \ldots, R_n) \in \mathbb{R}^n_{\geqslant 0} : \sum_{i=1}^n R_i \leqslant C\}$

Estimated process At a sampling time: $\hat{X}_i(t) = X_i(t)$. At other times: $d\hat{X}_i(t) = a_i \hat{X}_i(t)dt$

System Performance and Optimization Problem

Mean-square error $M_i(R_i) = R_i \int_0^{1/R_i} \left(X_i(t) - \hat{X}_I(t)\right)^2 dt$ when sensor i is sampling at rate R_i.

Example If the noise process is a Wiener process with variance σ_i^2, then the state process is a Gauss-Markov (or Ornstein-Uhlenbeck) process, and $M_i(R_i) = \frac{\sigma_i^2}{2a_i}\left[\frac{e^{2a_i/R_i - 1}}{2a_i/R_i} - 1\right]$.

System Performance and Optimization Problem

Mean-square error $M_i(R_i) = R_i \int_0^{1/R_i} \left(X_i(t) - \hat{X}_I(t)\right)^2 dt$ when sensor i is sampling at rate R_i.

Example If the noise process is a Wiener process with variance σ_i^2, then the state process is a Gauss-Markov (or Ornstein-Uhlenbeck) process, and $M_i(R_i) = \frac{\sigma_i^2}{2a_i}\left[\frac{e^{2a_i/R_i - 1}}{2a_i/R_i} - 1\right]$.

Assumptions

(A1) For any sensor i and rate $R_i > 0$, $M_i(R_i)$ is strictly decreasing and convex in R_i.

(A2) $M_i(R_i)$ is twice differentiable and there exists a positive constant c_i such that

$M_i''(R_i) \geqslant c_i$ for all $R_i > 0$.

System Performance and Optimization Problem

Mean-square error $M_i(R_i) = R_i \int_0^{1/R_i} \left(X_i(t) - \hat{X}_I(t)\right)^2 dt$ when sensor i is sampling at rate R_i.

Example If the noise process is a Wiener process with variance σ_i^2, then the state process is a Gauss-Markov (or Ornstein-Uhlenbeck) process, and $M_i(R_i) = \dfrac{\sigma_i^2}{2a_i}\left[\dfrac{e^{2a_i/R_i - 1}}{2a_i/R_i} - 1\right]$.

Assumptions

(A1) For any sensor i and rate $R_i > 0$, $M_i(R_i)$ is strictly decreasing and convex in R_i.

(A2) $M_i(R_i)$ is twice differentiable and there exists a positive constant c_i such that $M_i''(R_i) \geqslant c_i$ for all $R_i > 0$.

Problem formulation Find rate $(R_1, \dots, R_n) \in \mathbb{R}^n_{\geqslant 0}$ to $\min \sum_{i=1}^{n} M_i(R_i)$ such that $\sum_{i=1}^{n} R_i \leqslant C$.

Solution approach

Proposition Under assumptions (A1) and (A2), the optimization problem has a unique solution which we denote by $\mathbf{R}^* = (R_1^*, \ldots, R_n^*)$.

Solution approach

Proposition Under assumptions (A1) and (A2), the optimization problem has a unique solution which we denote by $\mathbf{R}^* = (R_1^*, \ldots, R_n^*)$.

How do we find $\mathbf{R}^*$ in a distributed manner?

Distributed Solution via Dual Decomposition

Primal Problem

$$\min_{\mathbf{R}^* \in \mathbb{R}^n_{\geqslant 0}} \sum_{i=1}^{n} M_i(R_i)$$

$$\text{s.t.} \sum_{i=1}^{n} R_i \leqslant C$$

Distributed Solution via Dual Decomposition

Primal Problem

$$\min_{\mathbf{R}^* \in \mathbb{R}^n_{\geqslant 0}} \sum_{i=1}^{n} M_i(R_i)$$

$$\text{s.t. } \sum_{i=1}^{n} R_i \leqslant C$$

Lagrangian Dual

$$\min_{\lambda \in \mathbb{R}_{\geqslant 0}} L(\mathbf{R}, \lambda)$$

$$\text{where } L(\mathbf{R}, \lambda) = \sum_{i=1}^{n} \left[M_i(R_i) + \lambda R_i \right] - \lambda C$$

Distributed Solution via Dual Decomposition

Primal Problem

$$\min_{\mathbf{R}^* \in \mathbb{R}^n_{\geqslant 0}} \sum_{i=1}^{n} M_i(R_i)$$

$$\text{s.t.} \sum_{i=1}^{n} R_i \leqslant C$$

Lagrangian Dual

$$\min_{\lambda \in \mathbb{R}_{\geqslant 0}} L(\mathbf{R}, \lambda)$$

$$\text{where } L(\mathbf{R}, \lambda) = \sum_{i=1}^{n} \left[M_i(R_i) + \lambda R_i \right] - \lambda C$$

Dual Decomposition

Decomposes into two parts: Network and Sensor i

[Kelly et al 1998]
[Low Lapsley 1999]
[Chiang et al 2007]

Distributed Solution via Dual Decomposition

Primal Problem

$$\min_{\mathbf{R}^* \in \mathbb{R}^n_{\geqslant 0}} \sum_{i=1}^{n} M_i(R_i)$$

$$\text{s.t.} \sum_{i=1}^{n} R_i \leqslant C$$

Lagrangian Dual

$$\min_{\lambda \in \mathbb{R}_{\geqslant 0}} L(\mathbf{R}, \lambda)$$

$$\text{where } L(\mathbf{R}, \lambda) = \sum_{i=1}^{n} \left[M_i(R_i) + \lambda R_i \right] - \lambda C$$

Dual Decomposition — Decomposes into two parts: Network and Sensor i

[Kelly et al 1998]
[Low Lapsley 1999]
[Chiang et al 2007]

Syncronous Algorithm

- Network starts with a guess λ_0.
- At each iteration $k = 0, 1, \ldots$

At each sensor i

Pick $R_{i,k}$ to min $M_i(R_i) + \lambda_k R_{i,k}$

At the network

$$\lambda_{k+1} = \left[\lambda_k - \alpha_k \left(C - \sum_{i=1}^{n} R_{i,k} \right) \right]^+$$

Properties of synchronous algorithm

Theorem 1 Under (A1) and (A2), for any initial guess λ_0 and appropriately chosen step sizes α_k,

$$\lim_{k\to\infty} \mathbf{R}_k := \lim_{k\to\infty} (R_{1,k}, \ldots, R_{n,k}) = \mathbf{R}^*.$$

Properties of synchronous algorithm

Theorem 1 Under (A1) and (A2), for any initial guess λ_0 and appropriately chosen step sizes α_k,

$$\lim_{k\to\infty} \mathbf{R}_k := \lim_{k\to\infty} (R_{1,k}, \ldots, R_{n,k}) = \mathbf{R}^*.$$

Implementation The synchronous algorithm can be implemented as part of the initial handshaking protocol when the sensors come online.

Drawbacks

- Large signaling overhead.
- Algorithm needs to be rerun when:
  - a sensor leaves, or
  - a new sensor comes on board, or
  - the network capacity changes.

Properties of synchronous algorithm

Theorem 1 Under (A1) and (A2), for any initial guess λ_0 and appropriately chosen step sizes α_k,

$$\lim_{k\to\infty} \mathbf{R}_k := \lim_{k\to\infty} (R_{1,k}, \ldots, R_{n,k}) = \mathbf{R}^*.$$

Implementation The synchronous algorithm can be implemented as part of the initial handshaking protocol when the sensors come online.

Drawbacks

- Large signaling overhead.
- Algorithm needs to be rerun when:
  - a sensor leaves, or
  - a new sensor comes on board, or
  - the network capacity changes.

> Salient feature: The network doesn't need to know $R_{i,k}$. It only needs an estimate of $\sum R_i$, which it can infer from the received packets.

Asynchronous algorithm for choosing sampling rates

At the network

Initialize $\lambda > 0$

Upon event $\langle$new packet received$\rangle$ **do**

- Estimate received sum rate $\hat{C}_k$ based on packets received in a sufficiently large sliding window of time.
- $\lambda_{k+1} = \left[\lambda_k - \alpha_k\left(C - \hat{C}_k\right)\right]^+$.
- Broadcast λ_{k+1}

Asynchronous algorithm for choosing sampling rates

At the network

Initialize $\lambda > 0$

Upon event $\langle$new packet received$\rangle$ **do**

- Estimate received sum rate $\hat{C}_k$ based on packets received in a sufficiently large sliding window of time.
- $\lambda_{k+1} = \left[\lambda_k - \alpha_k\left(C - \hat{C}_k\right)\right]^+$.
- Broadcast λ_{k+1}

At each sensor

Upon event $\langle$initialize$\rangle$ or $\langle$take new sample$\rangle$ **do**

- Observe λ
- Pick R_i to min $M_i(R_i) + \lambda R_i$
- Set next sampling time $=$ current time $+\frac{1}{R_i}$.

Properties of asynchronous algorithm

Assumption (A3) The time between the consecutive samples is bounded.

Theorem 2 Under (A1)–(A3), for any initial guess λ_0 and appropriately chosen step sizes α_k,

$$\lim_{k\to\infty} \mathbf{R}_k := \lim_{k\to\infty} (R_{1,k}, \ldots, R_{n,k}) = \mathbf{R}^*.$$

Moreover, if the synchronous and the asynchronous algorithms use the same learning rates $\{\alpha_k\}_{k\geqslant 0}$, then the corresponding Lagrange multipliers converge to the same value.

Properties of asynchronous algorithm

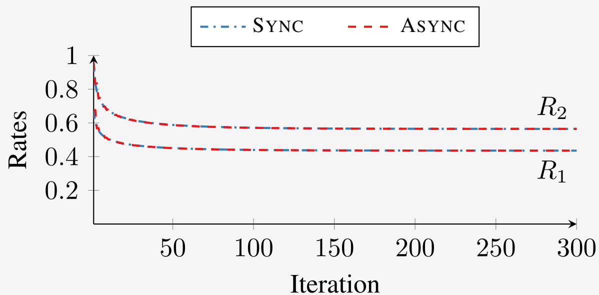
Assumption (A3) The time between the consecutive samples is bounded.

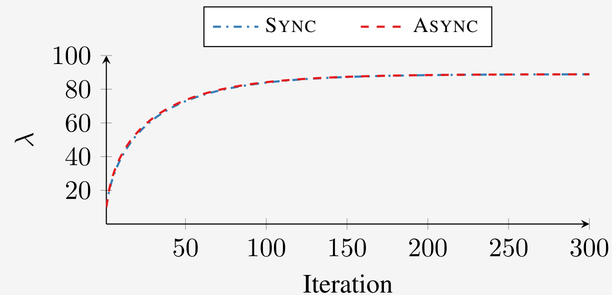
Theorem 2 Under (A1)–(A3), for any initial guess λ_0 and appropriately chosen step sizes α_k,

$$\lim_{k\to\infty} \mathbf{R}_k := \lim_{k\to\infty} (R_{1,k}, \ldots, R_{n,k}) = \mathbf{R}^*.$$

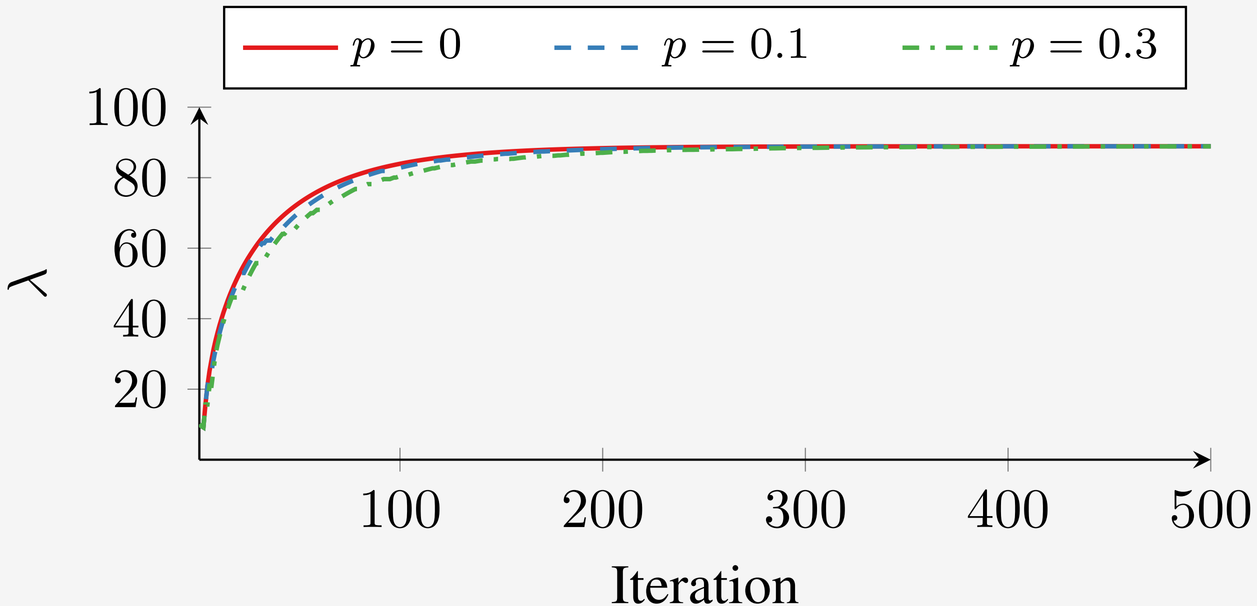
Moreover, if the synchronous and the asynchronous algorithms use the same learning rates $\{\alpha_k\}_{k\geqslant 0}$, then the corresponding Lagrange multipliers converge to the same value.

Example 2 sensors: GaussMarkov$(1,1)$ and GaussMarkov$(1,2)$. Network capacity $C = 1$.





Robustness to packet drops and delays



Illustrative example

Changing network conditions

- Sensors arrive according to a Poisson process and stay in the system for an exponentially distributed amount of time.
- Sensor parameters are distributed randomly.
- Network capacity changes exogeneously.

Illustrative example

Changing network conditions

- Sensors arrive according to a Poisson process and stay in the system for an exponentially distributed amount of time.
- Sensor parameters are distributed randomly.
- Network capacity changes exogeneously.

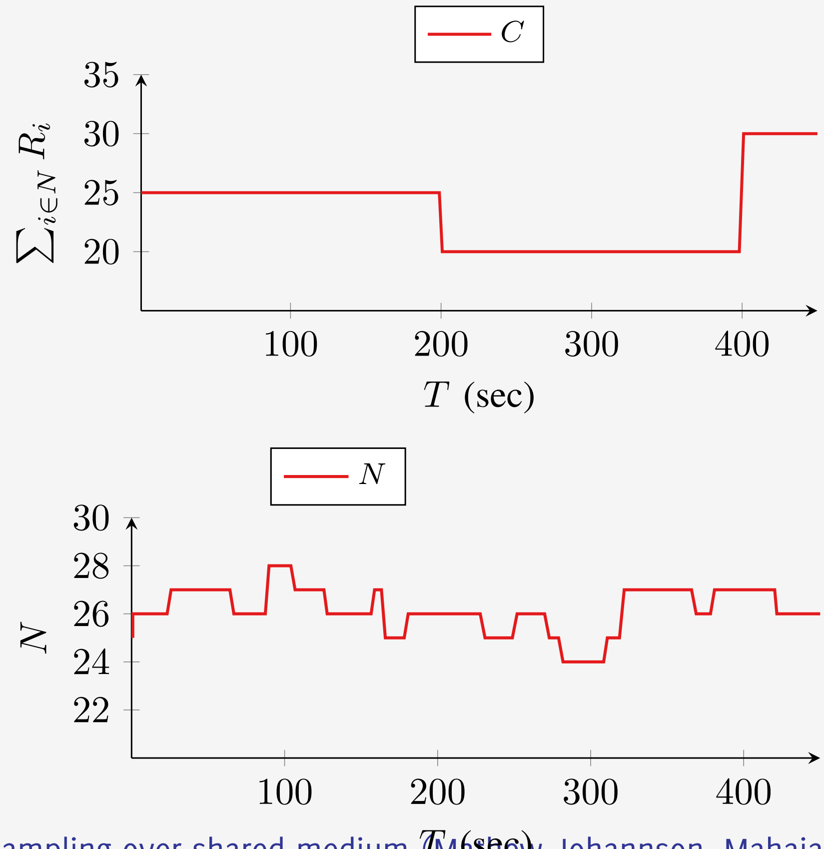
Network

- Network is not aware of the number of sensors.
- Adapts λ according to the asynchronous algorithm
- Broadcasts the value of λ.

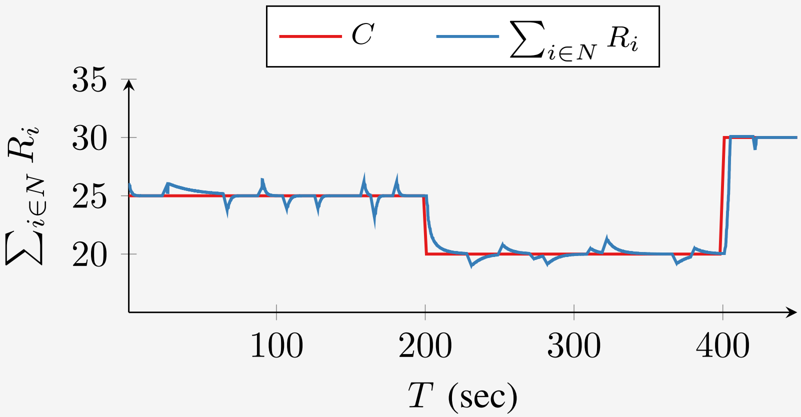
Sensors

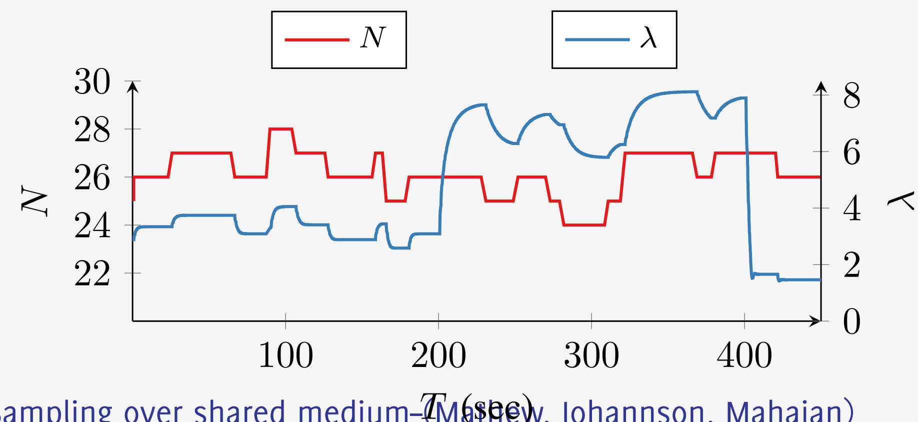
- Sensors don't know the network capacity.
- Run the asynchronous algorithm to adapt rate R_i.

Illustrative example: System parameters vs time

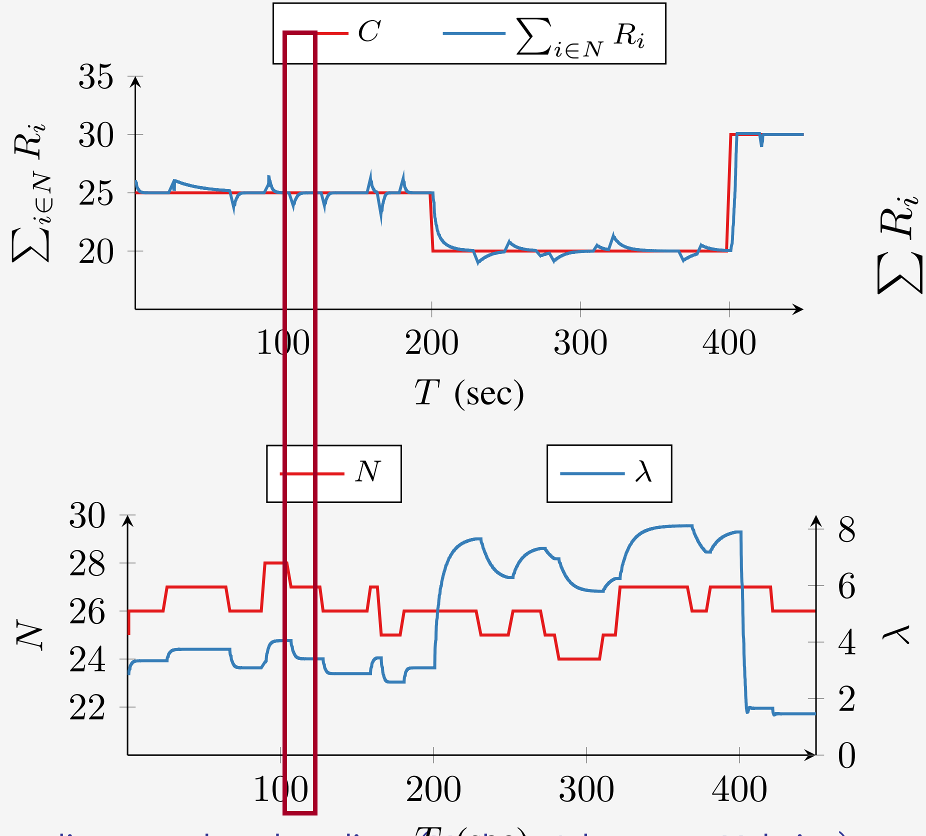


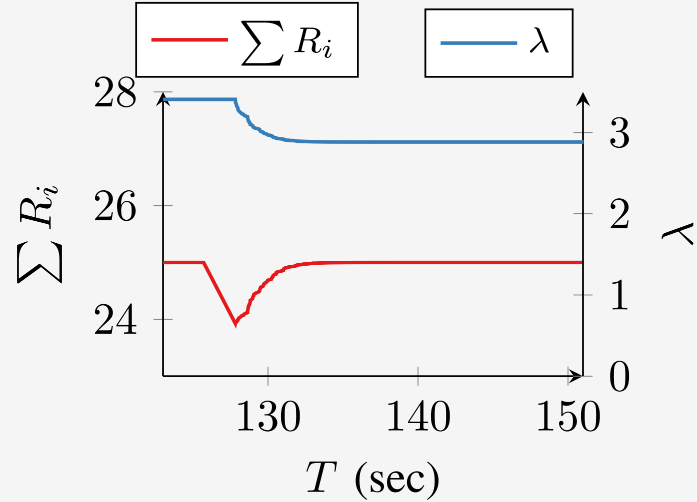
Illustrative example: System parameters vs time





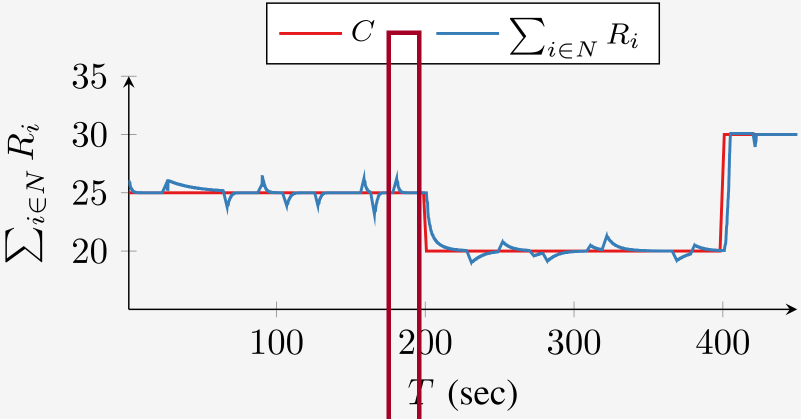
Illustrative example: System parameters vs time

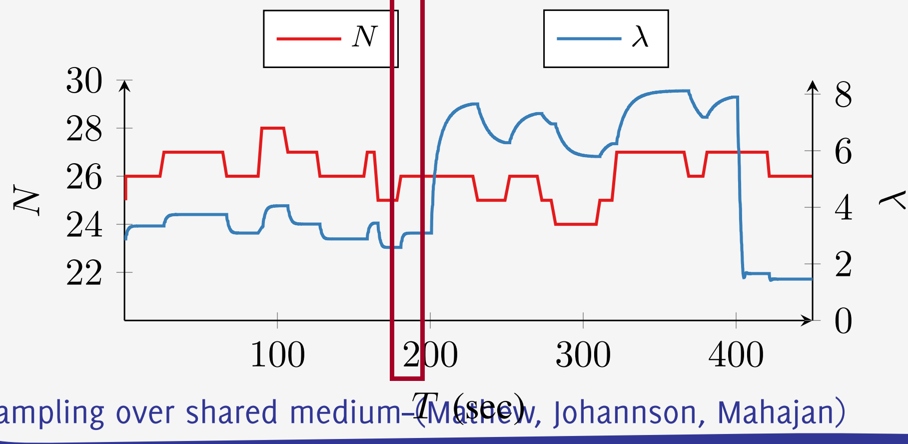


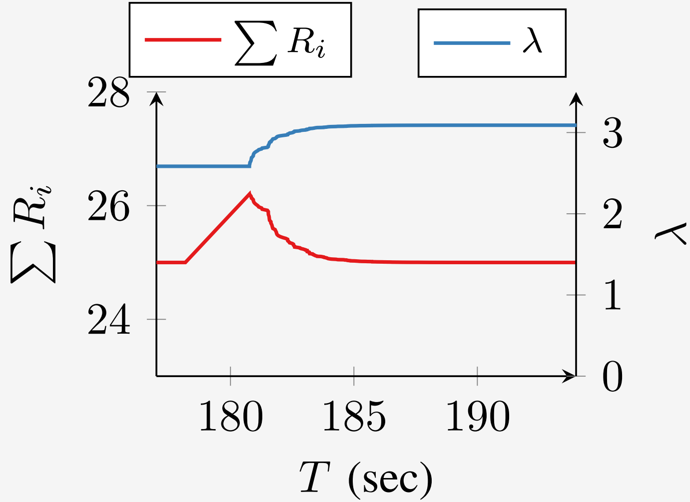


Sensor leaving

Illustrative example: System parameters vs time

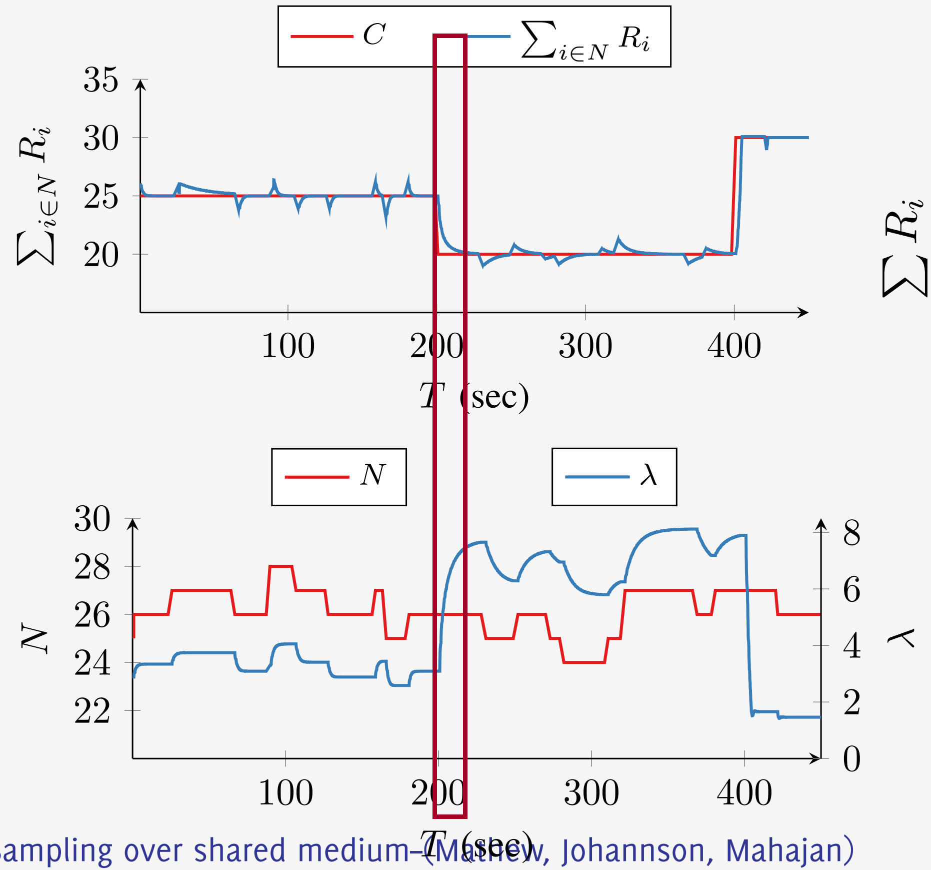


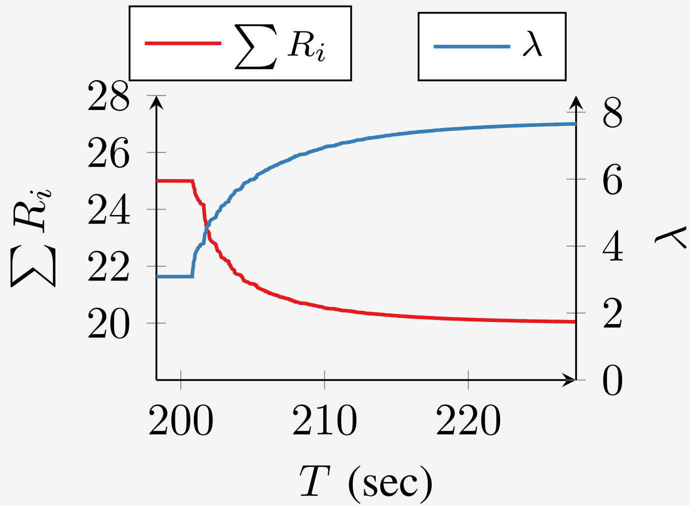




Sensor coming on board

Illustrative example: System parameters vs time





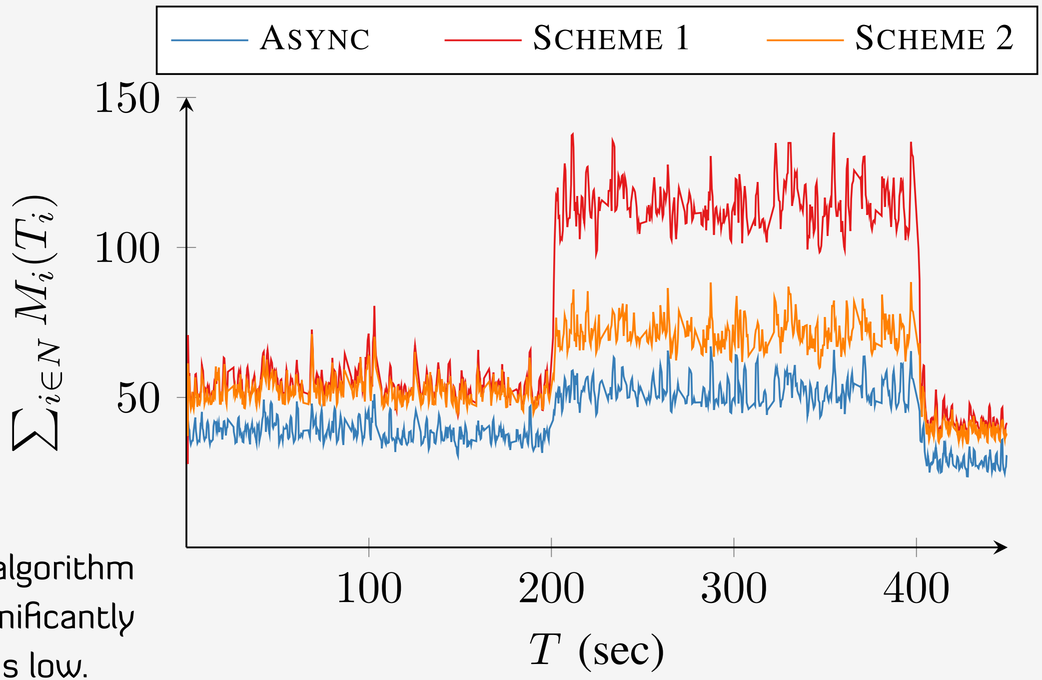
Network capacity changing

Comparison with baseline schemes

- Scheme 1: $R_i = C/30$.
- Scheme 2: $R_i = C/N(t)$.

Comparison with baseline schemes

- Scheme 1: $R_i = C/30$.
- Scheme 2: $R_i = C/N(t)$.

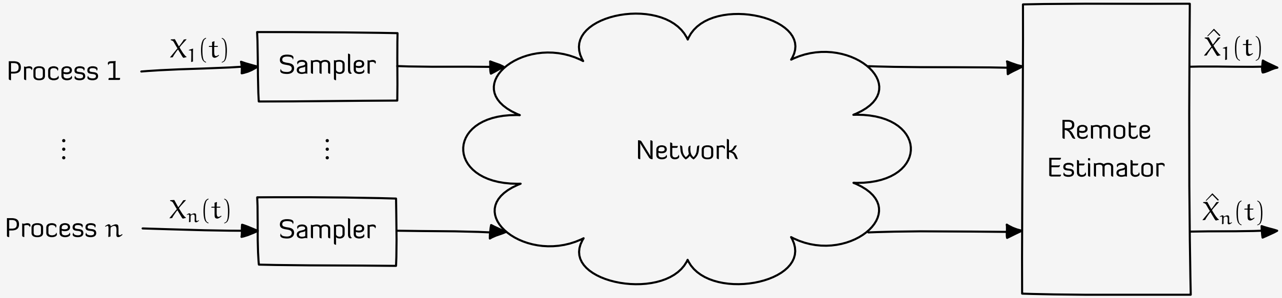


- Performance of Asynchronous algorithm is better than baseline, and significantly so when the network capacity is low.

Summary

Summary

System Model



Process dynamics	$dX_i(t) = a_i X_i(t)dt + dW_i(t)$. $\{W_i(t)\}_{t\geq 0}$ is stationary and indep across sensors.
Sampling process	Sensor i samples process i at rate $R_i = 1/T_i$.
Network	Rate region $\mathcal{R} = \left\{(R_1, \dots, R_n) \in \mathbb{R}^n_{\geq 0} : \sum_{i=1}^n R_i \leq C\right\}$
Estimated process	At a sampling time: $\hat{X}_i(t) = X_i(t)$. At other times: $d\hat{X}_i(t) = a_i \hat{X}_i(t)dt$

Summary

System Performance and Optimization Problem

Mean-square error $M_i(R_i) = R_i \int_0^{1/R_i} \left(X_i(t) - \hat{X}_I(t)\right)^2 dt$ when sensor i is sampling at rate R_i.

Example If the noise process is a Wiener process with variance σ_i^2, then the state process is a Gauss-Markov (or Ornstein-Uhlenbeck) process, and $M_i(R_i) = \frac{\sigma_i^2}{2a_i}\left[\frac{e^{2a_i/R_i - 1}}{2a_i/R_i} - 1\right]$.

Assumptions

(A1) For any sensor i and rate $R_i > 0$, $M_i(R_i)$ is strictly decreasing and convex in R_i.

(A2) $M_i(R_i)$ is twice differentiable and there exists a positive constant c_i such that $M_i''(R_i) \geqslant c_i$ for all $R_i > 0$.

Problem formulation Find rate $(R_1, \ldots, R_n) \in \mathbb{R}^n_{\geqslant 0}$ to $\min \sum_{i=1}^{n} M_i(R_i)$ such that $\sum_{i=1}^{n} R_i \leqslant C$.

Summary

System Model

System Performance and Optimization Problem

Distributed Solution via Dual Decomposition

Primal Problem

$$\min_{\mathbf{R}^* \in \mathbb{R}^n_{\geqslant 0}} \sum_{i=1}^{n} M_i(R_i)$$

$$\text{s.t.} \sum_{i=1}^{n} R_i \leqslant C$$

Lagrangian Dual

$$\min_{\lambda \in \mathbb{R}_{\geqslant 0}} L(\mathbf{R}, \lambda)$$

$$\text{where } L(\mathbf{R}, \lambda) = \sum_{i=1}^{n} \left[M_i(R_i) + \lambda R_i\right] - \lambda C$$

Dual Decomposition — Decomposes into two parts: Network and Sensor i

[Kelly et al 1998]
[Low Lapsley 1999]
[Chiang et al 2007]

Syncronous Algorithm

- Network starts with a guess λ_0.
- At each iteration $k = 0, 1, \ldots$

At each sensor i

Pick $R_{i,k}$ to min $M_i(R_i) + \lambda_k R_{i,k}$

At the network

$$\lambda_{k+1} = \left[\lambda_k - \alpha_k \left(C - \sum_{i=1}^{n} R_{i,k}\right)\right]^+$$

Summary

System Model

System Performance and Optimization Problem

Distributed Solution via Dual Decomposition

Asynchronous algorithm for choosing sampling rates

At the network

Initialize $\lambda > 0$

Upon event $\langle$new packet received$\rangle$ **do**

- Estimate received sum rate $\hat{C}_k$ based on packets received in a sufficiently large sliding window of time.
- $\lambda_{k+1} = \left[\lambda_k - \alpha_k\left(C - \hat{C}_k\right)\right]^+$.
- Broadcast λ_{k+1}

At each sensor

Upon event $\langle$initialize$\rangle$ or $\langle$take new sample$\rangle$ **do**

- Observe λ
- Pick R_i to min $M_i(R_i) + \lambda R_i$
- Set next sampling time $=$ current time $+ \dfrac{1}{R_i}$.

Summary

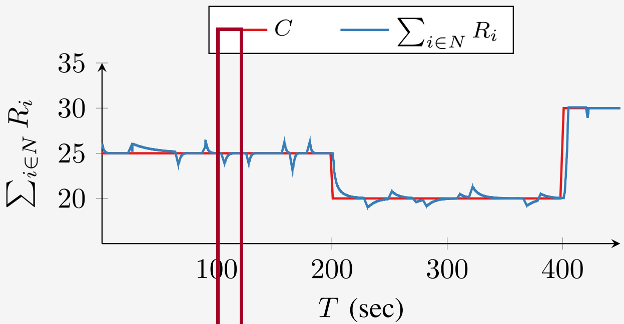
System Model

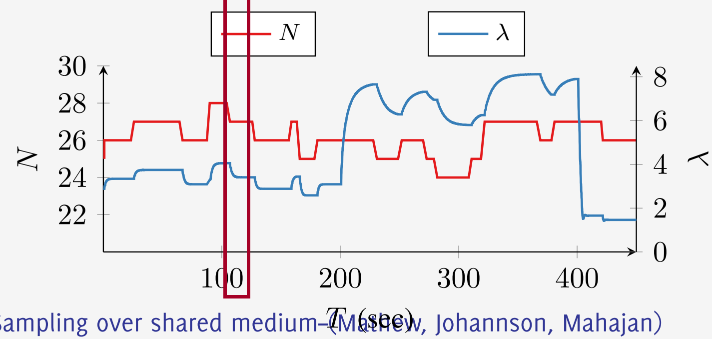
System Performance and Optimization Problem

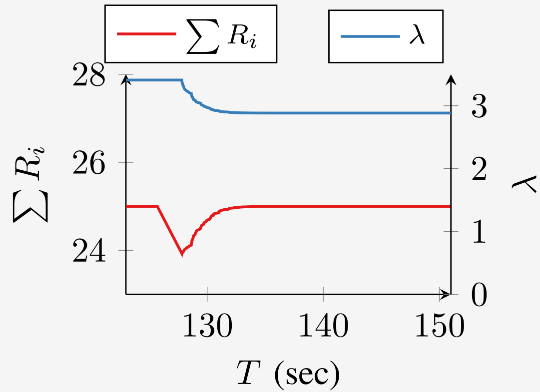
Distributed Solution via Dual Decomposition

Asynchronous algorithm for choosing sampling rates

Illustrative example: System parameters vs time







Sensor leaving

Summary

System Model

System Performance and Optimization Problem

Distributed Solution via Dual Decomposition

Asynchronous algorithm for choosing sampling rates

Illustrative example—System parameters vs time

Conclustion

The asynchronous event-driven algorithm can adapt to slowly varying network conditions in a distributed manner. Asymptotically, the algorithm converges to the optimal rates.

The sensors and the estimators don't need synchronous clocks!

Robust to packet drops, delays, and slow variation in system parameters.

Dual decomposition does not ensure that the constraint $\sum R_i \leqslant C$ is satisfied at all iterations. In practice, violation of this constraint will lead to congestion. To understand its impact, we need to consider a more elaborate network model where congestion leads to delay.