

# BART

## Filter-based Estimation of End-to-End Available Bandwidth

---

EvaluNet workshop  
SICS, Stockholm, 19 December 2006

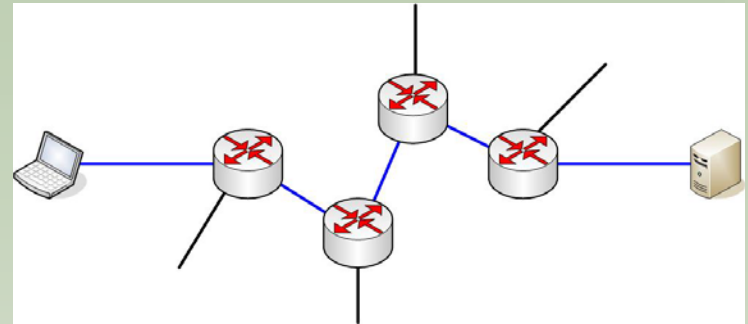
Svante Ekelin<sup>1,4</sup>, Martin Nilsson<sup>2</sup>, Erik Hartikainen<sup>1,3</sup>, Andreas Johnsson<sup>4</sup>,  
Jan-Erik Mångs<sup>1</sup>, Bob Melander<sup>1,4</sup>, Christofer Flinta<sup>1</sup>, Mats Björkman<sup>4</sup>

<sup>1</sup> Ericsson Research, <sup>2</sup> Swedish Institute of Computer Science,  
<sup>3</sup> Linköping University, <sup>4</sup> Mälardalen University

# outline

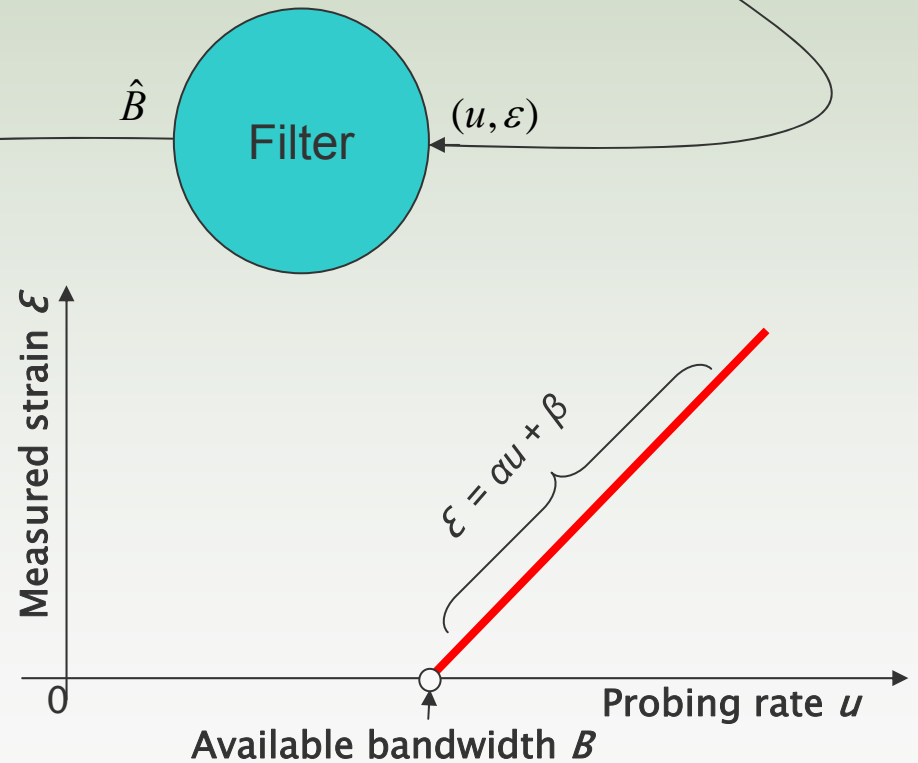
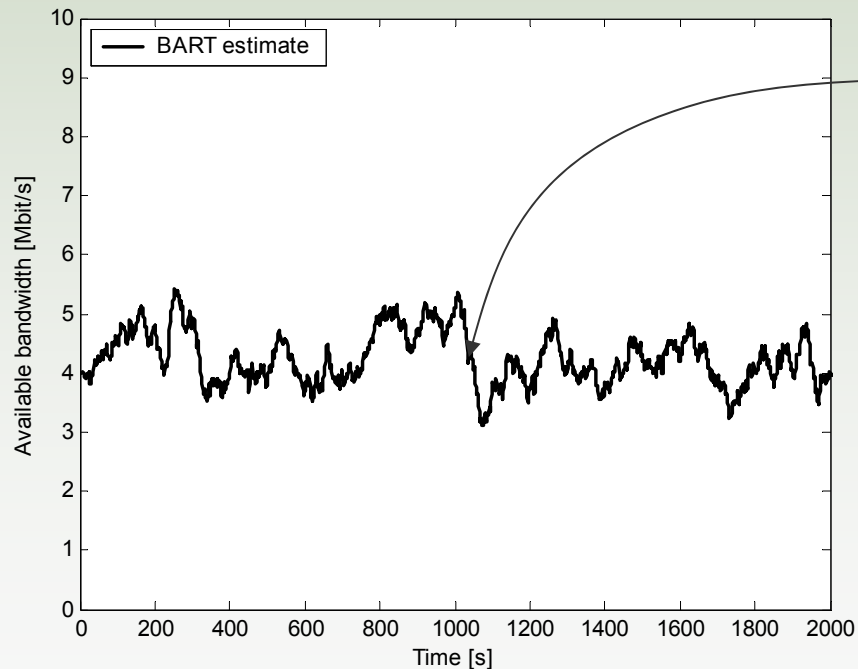
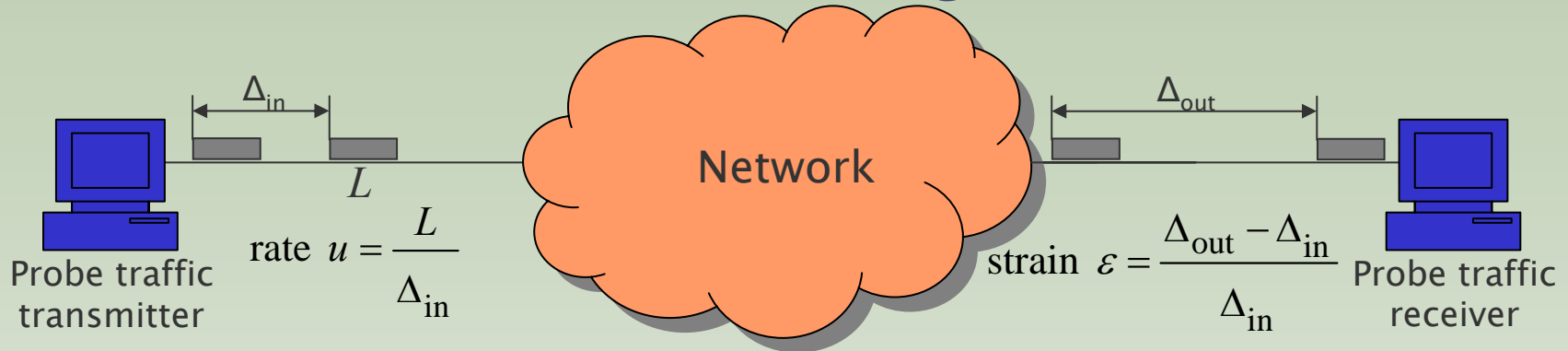
- ❖ problem statement
- ❖ executive summary
- ❖ available bandwidth basics
- ❖ filtering basics
- ❖ filtering applied to bandwidth estimation - BART
- ❖ temporal structure
- ❖ adaptivity to change of time scale
- ❖ empirical validation

# problem statement



- ❖ **how can we monitor end-to-end available bandwidth when we**
  - don't have access to information from intermediate network nodes?
  - don't even know the path (link capacities etc)?
  
- ❖ **why would we want to monitor available bandwidth?**
  - SLA verification
  - network performance monitoring
  - fault detection
  - traffic management
  - adaptive applications
  - congestion control
  - server/peer selection
  - multihoming (interface selection)
  - ...

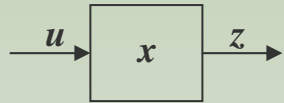
# executive summary (1/2)



# executive summary (2/2)

We developed **BART**, a method for **estimation of time-dependent available bandwidth**.

BART uses a **filtering method** to produce an updated estimate for each sampling.



$$x_k = f(x_{k-1}) + w_k$$

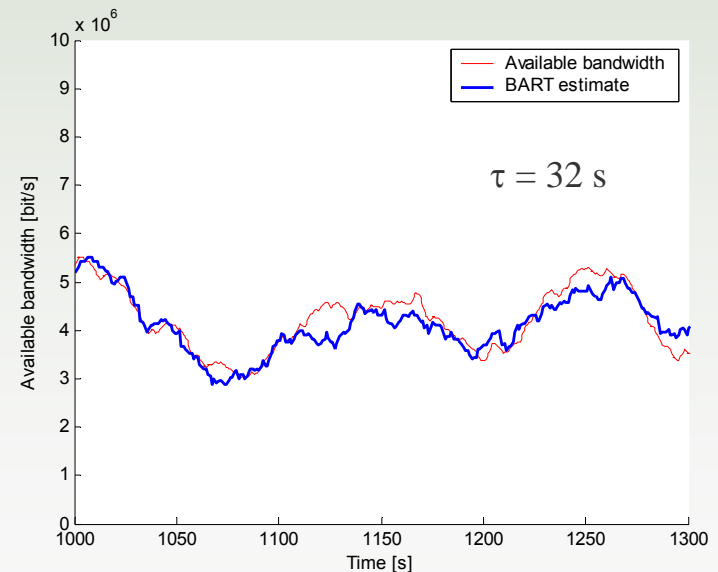
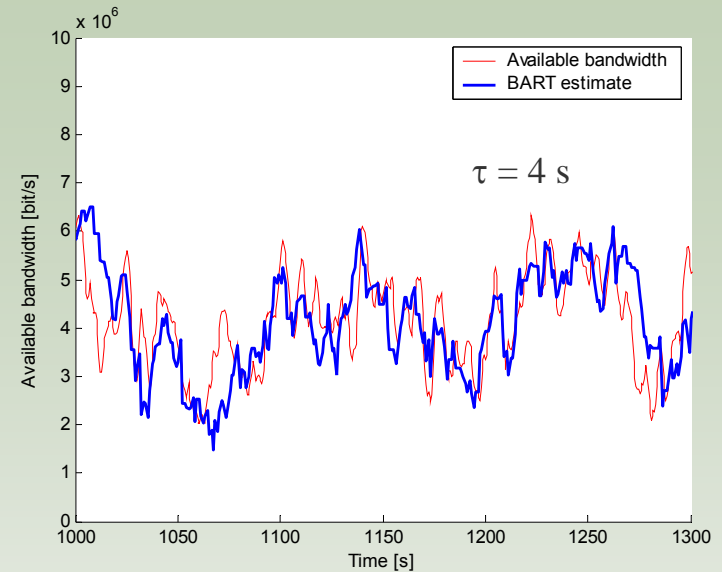
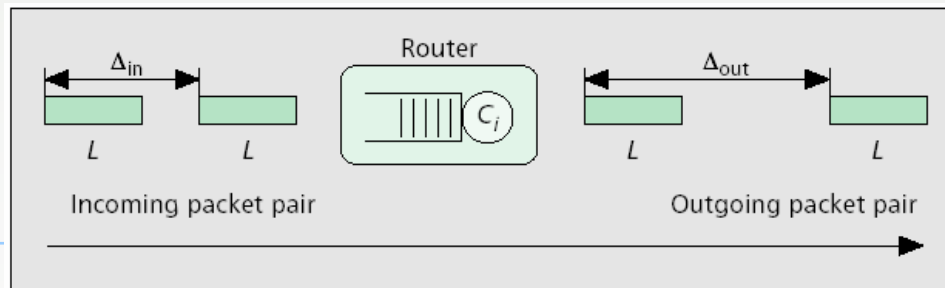
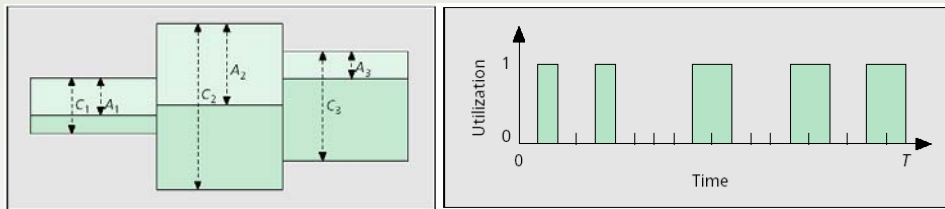
$$z_k = h(x_k) + v_k$$

Allows **recursive estimation** of system state:

$$\hat{x}_k = g(\hat{x}_{k-1}, z_k)$$

Benefits: **fast** (configurably)  
**accurate** (reasonably)  
**tunable to desired time scale**

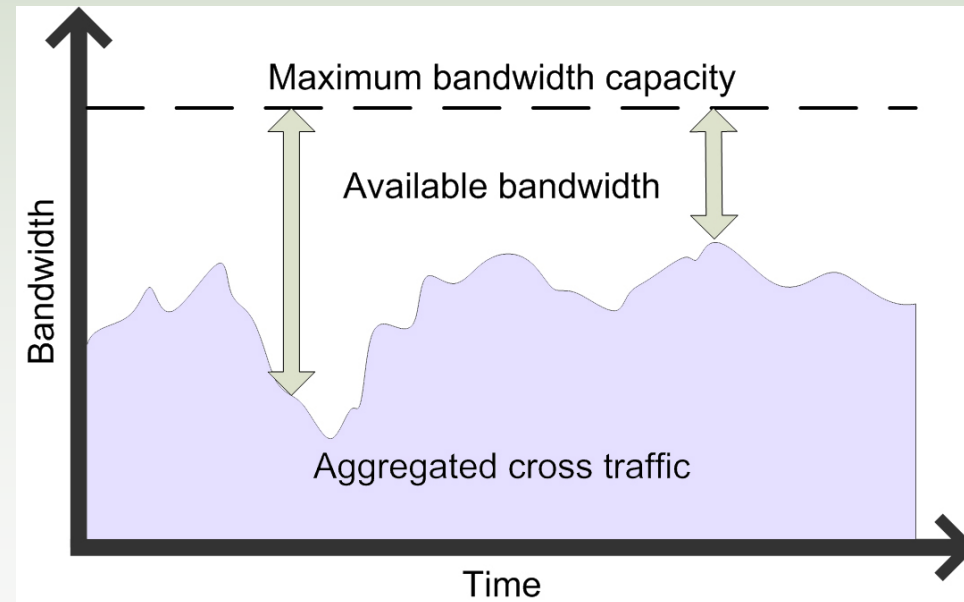
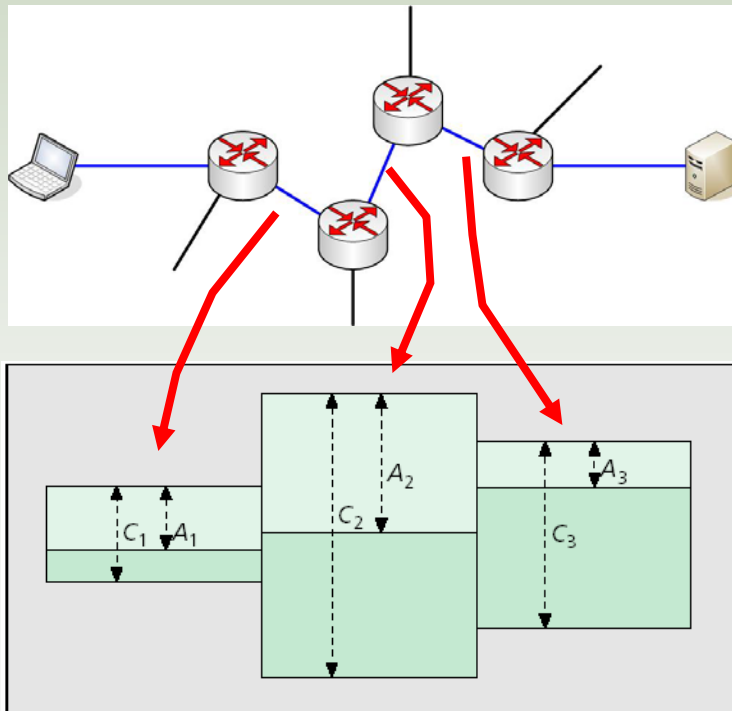
Artefacts: **C++ prototype implementation**  
**several papers**



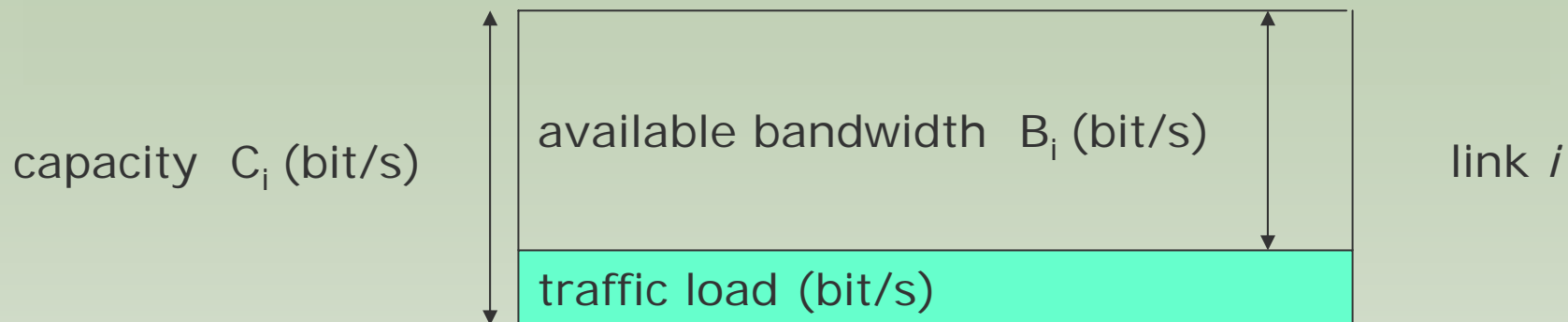
# What is available bandwidth?

the available bandwidth (unused traffic capacity) may be defined:

- ❖ link available bandwidth = link capacity - link traffic load
- ❖ path available bandwidth = smallest available bandwidth of links along the path



# bandwidth-related parameters



Typically for a fixed link, the capacity is constant, whereas traffic load (and hence available bandwidth) varies in time.

Link parameters of interest:

capacity  $C_i$  and available bandwidth  $B_i = B_i(t)$

**Path parameters of interest:**

( path capacity

$$C_{\text{path}} = \min C_i$$

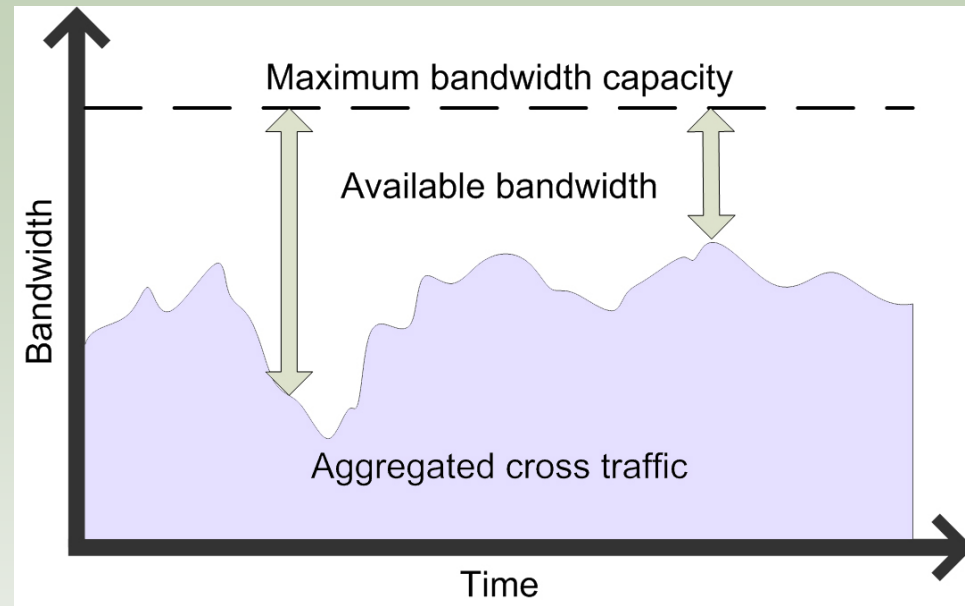
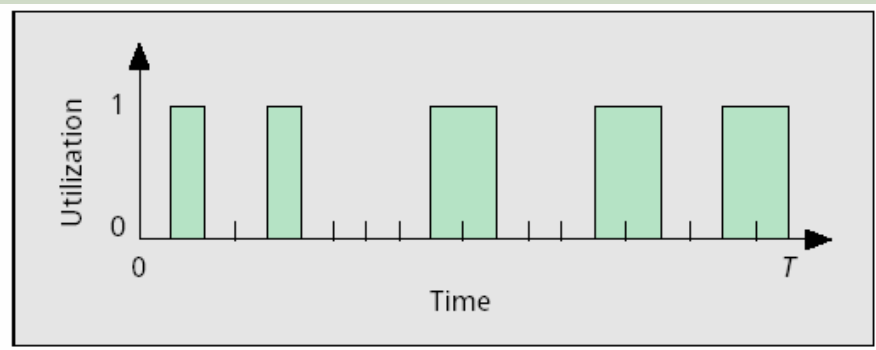
**path available bandwidth**

$$\mathbf{B} = \min B_i = \text{abw of tight link}$$

**tight link capacity**

**C**

# traffic averaging time scale



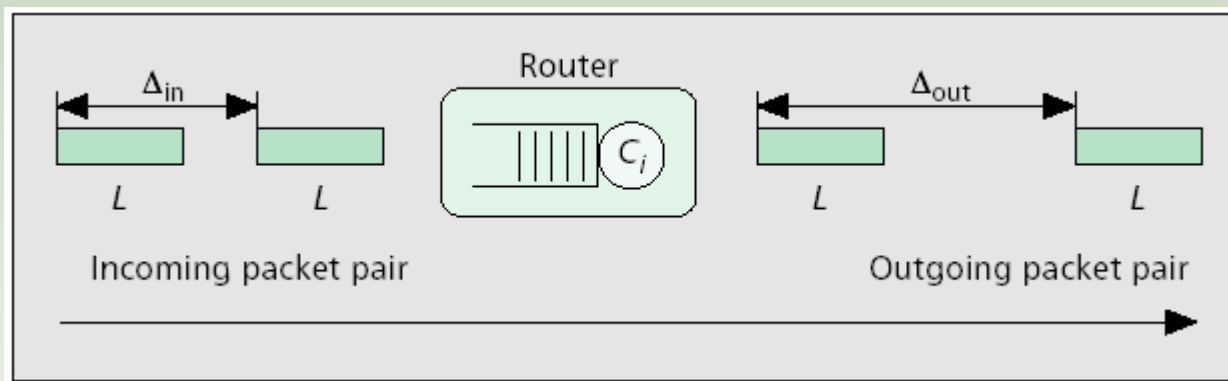
Instantaneous utilization of a link is either 0 or 100% .

For practical purposes, traffic load and thus available bandwidth are defined relative to some arbitrary time resolution  $T \gg$  packet time scale.

$$\text{traffic load} = \frac{\text{number of bits during time } T}{T}$$

# congestion => time dispersion

We probe the network path using pairs of probe packets.



If the local probing rate  $u = L / \Delta_{in}$  is larger than the available bandwidth, there is (transient) congestion.

As a congestion measure, we use the relative time dispersion, or "strain",  $\varepsilon = (\Delta_{out} - \Delta_{in}) / \Delta_{in}$

$$[ \varepsilon = u_{in} / u_{out} - 1 ]$$

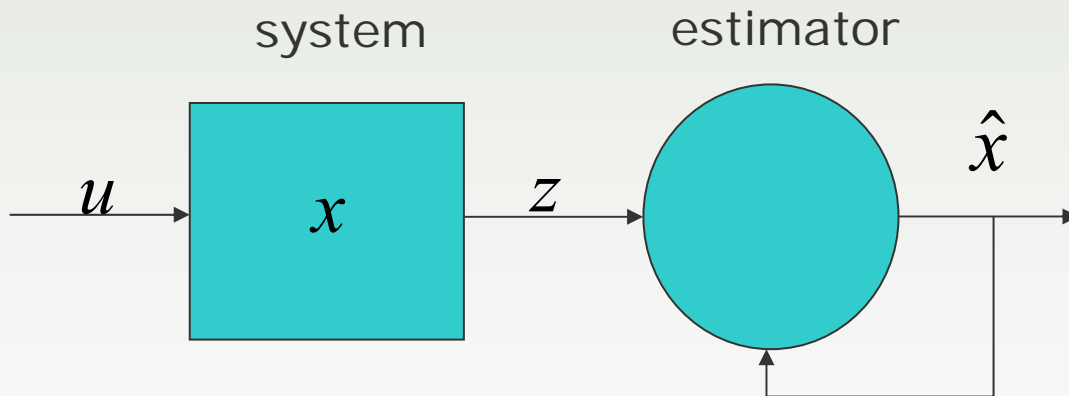
# filtering approach: recursive estimation

## ❖ framework

- We want to estimate the state  $x$  of a system without having direct access. However, we can observe an output  $z$ , which depends on the state, and we can influence the system with a control input  $u$ .
- We keep a current state estimate, and for each new sampling we calculate the updated estimate from the previous estimate and the new measurement.

## ❖ estimator algorithm depends on

- **system model** describing how the system state  $x$  evolves in time
- **measurement model** describing how  $z$  depends on  $x$



estimator calculates:

$$\hat{x}_k = g(\hat{x}_{k-1}, z_k)$$

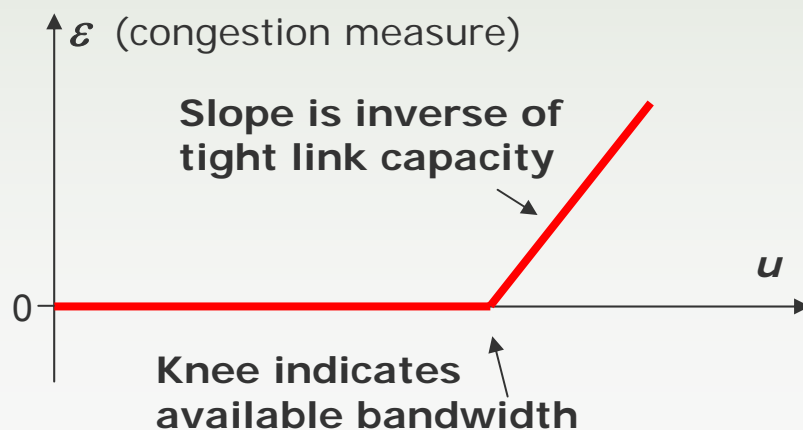
# modeling our problem

❖ the system to be estimated is the network path

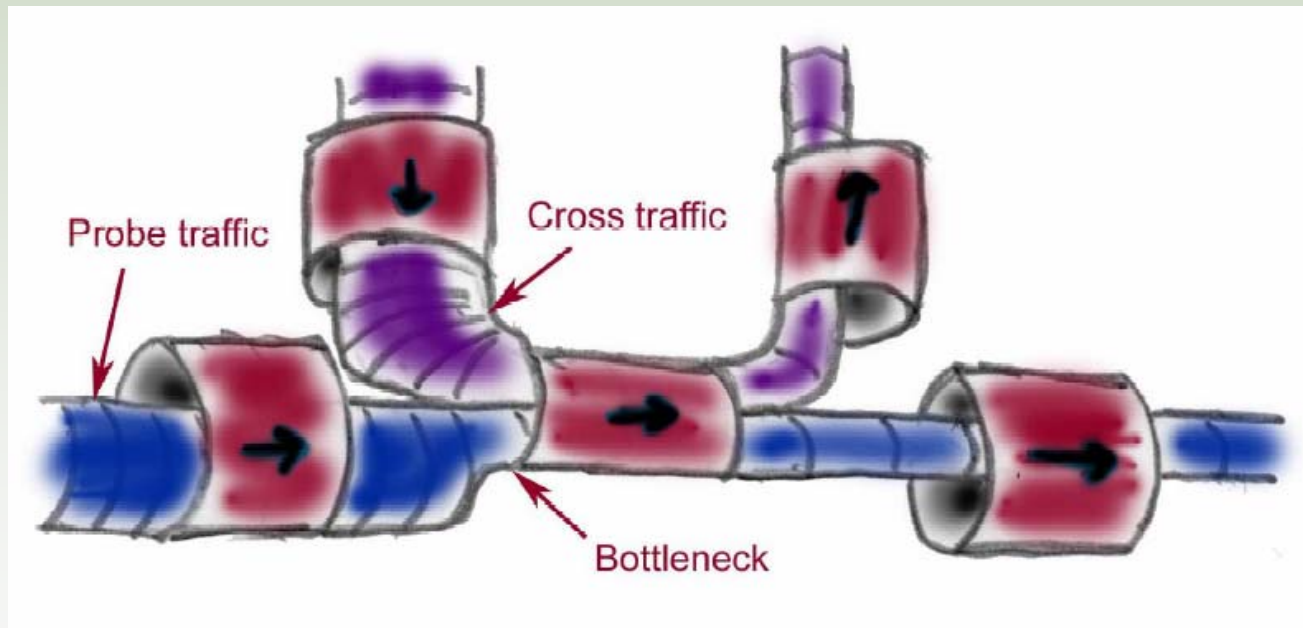


- system state potentially has very high dimensionality, if taking into account detailed static and dynamic characteristics  $s_i$  and  $d_i(t)$  for hop  $i = 1, \dots, N$
- but we want to avoid unnecessary complexity
- from previous bandwidth estimation methodology work:

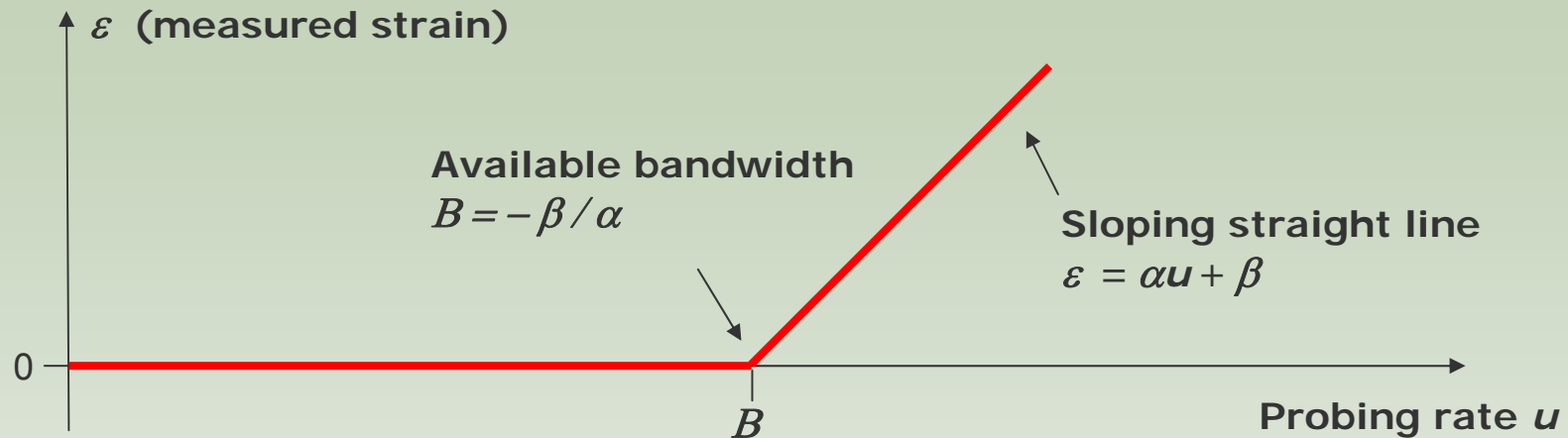
In a fluid model of traffic flow, one expects a piecewise linear dependence of the strain  $\varepsilon$  on the probing rate  $u$ . Further, only the tight hop contributes as long as there are not multiple congested hops.  
(Melander, 2000)



# proportional sharing



# the BART approach



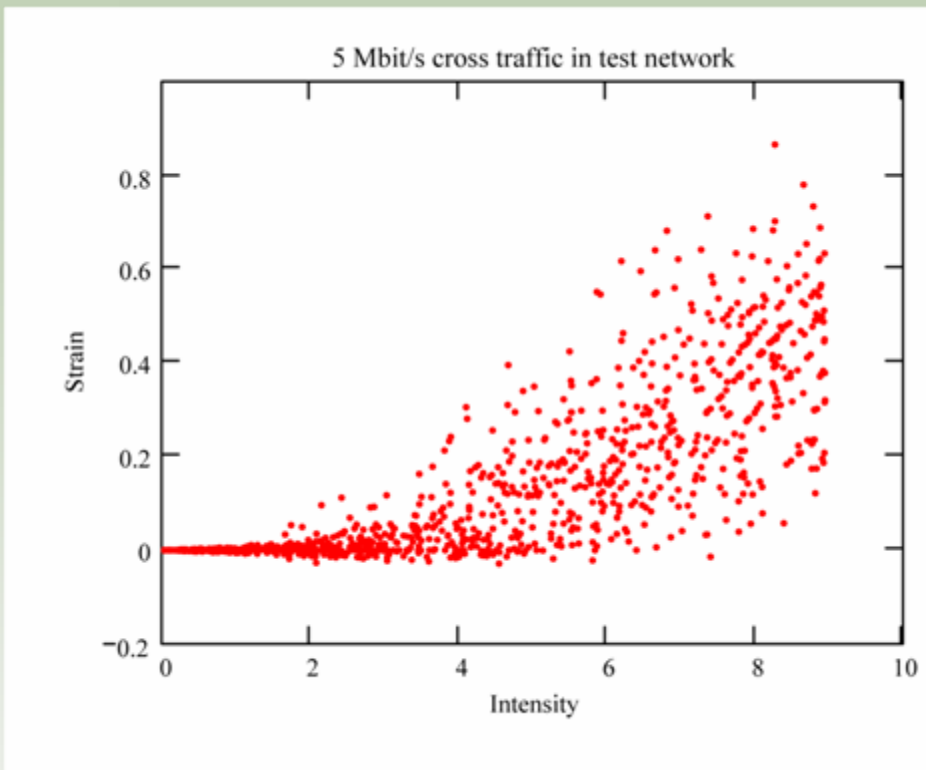
- this curve describes dominating contribution to strain
- the relevant aspects of the system state for our purpose are captured by the sloping straight line

## ❖ BART model:

- system described by two-dimensional state vector  $x = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$
- system evolution model :  $x_k = x_{k-1} + w_{k-1}$
- measurement model :  $\varepsilon_k = Hx_k + v_k$  where  $H = \begin{bmatrix} u & 1 \end{bmatrix}$ .

( $w$  and  $v$  are noise terms; for the sake of linearity we only consider the region  $u > B$ )

# what we actually see...



Each point represents a sampling using one probe packet train.

One sampling per second for approx. 1000 seconds.

Cross traffic is Poisson with long-time average intensity 5 Mb/s .

Tight link capacity is 10 Mb/s .

## Horizontal noise and vertical noise.

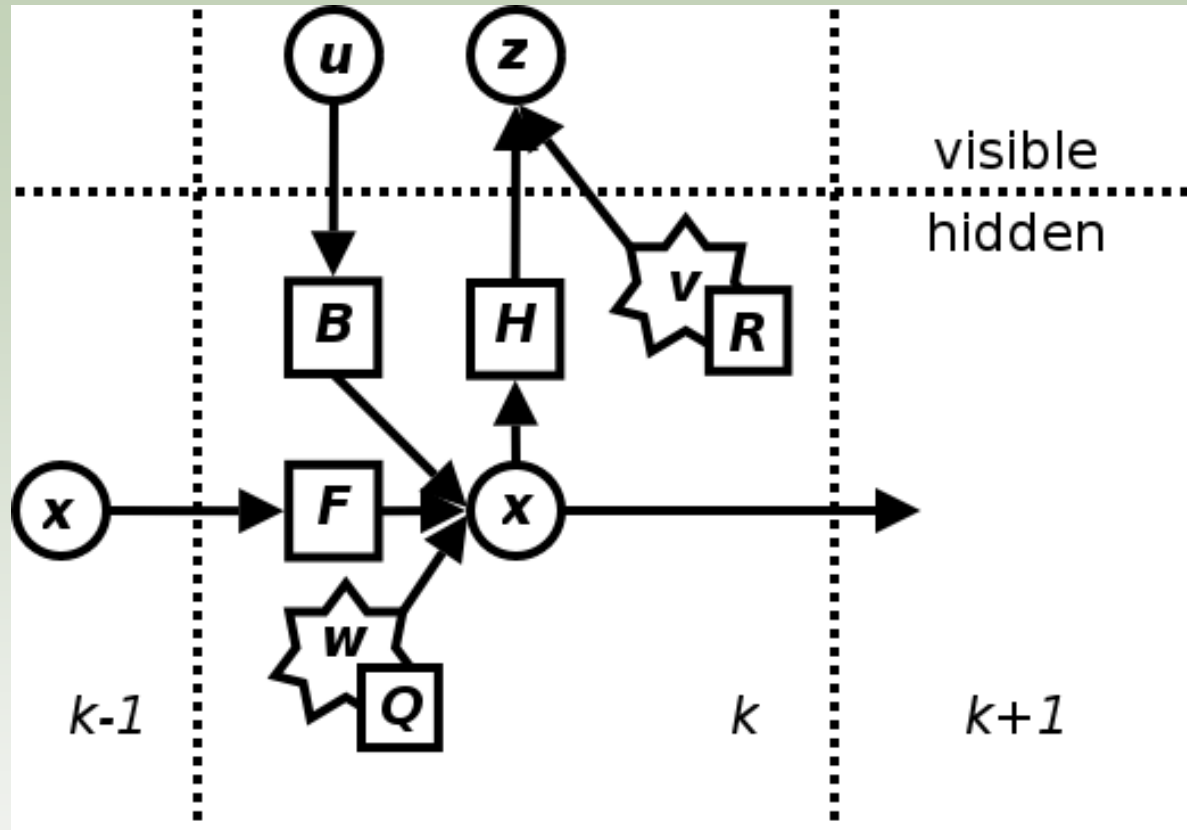
- ❖ horizontal spreading due to short-timescale variations in system state
  - ❖ cf. noise  $w$  in the system evolution model
- ❖ vertical spreading due to finite measurement precision
  - ❖ cf. noise  $v$  in the measurement model

We capture the real-time traffic characteristics by updating the estimate for each sample. This is achieved by Kalman filtering.

# Kalman filtering

- ❖ **Since we managed to express the model linearly, we can apply the Kalman filter!** (Kalman, 1960)
- ❖ **The Kalman filter is a special type of recursive estimator, with a host of desirable properties :**
  - easy to implement in a few lines of code
  - light-weight w.r.t. memory and CPU
  - tried and tested in many other engineering fields
    - trajectory tracking, computer graphics, analog signal control (phase-locked loop)...
  - proven to be optimal estimator under specific conditions
  - often well-performing even if these conditions are broken

# Kalman filter model

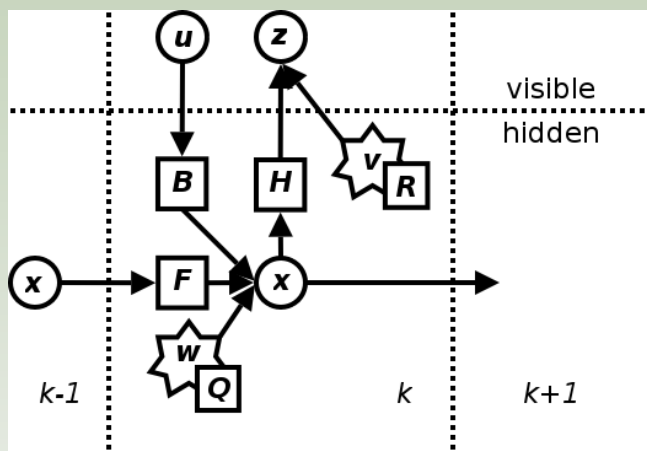


Generic model underlying the Kalman filter. Circles are [vectors](#), squares are [matrices](#), and stars represent [gaussian noise](#) with the associated [covariance matrix](#) at the lower right.  $x$  is the unknown system state,  $u$  the control input and  $z$  the measured output.

cf. [http://en.wikipedia.org/wiki/Kalman\\_filter](http://en.wikipedia.org/wiki/Kalman_filter)

# Kalman model equations

in general:



$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k$$

(system model)

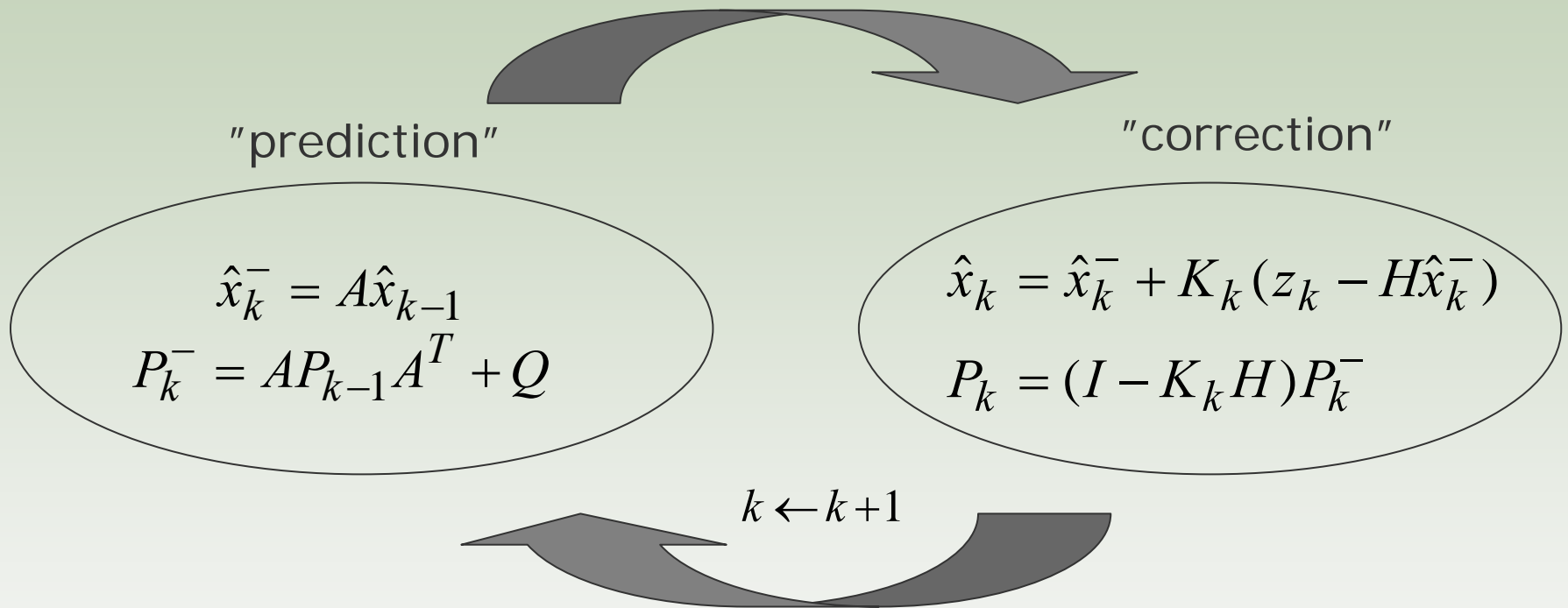
$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k$$

(measurement model)

as applied in BART:

- ❖ system state described by a two-dimensional state vector :  $x = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$
- ❖ system evolution described by the process  $x_k = x_{k-1} + w_{k-1}$ .
- ❖ measurement as function of system state described by  $\varepsilon_k = Hx_k + v_k$  where  $H = \begin{bmatrix} u & 1 \end{bmatrix}$ .
- ❖  $w$  and  $v$  are the process noise and measurement noise. Their covariances:  $Q$  and  $R$ .
  - ❖ (note: "process noise"  $w$  is really the "signal" from a network point of view, i.e.  $w = \Delta x$ , the change in system state due to fluctuations in cross traffic.)

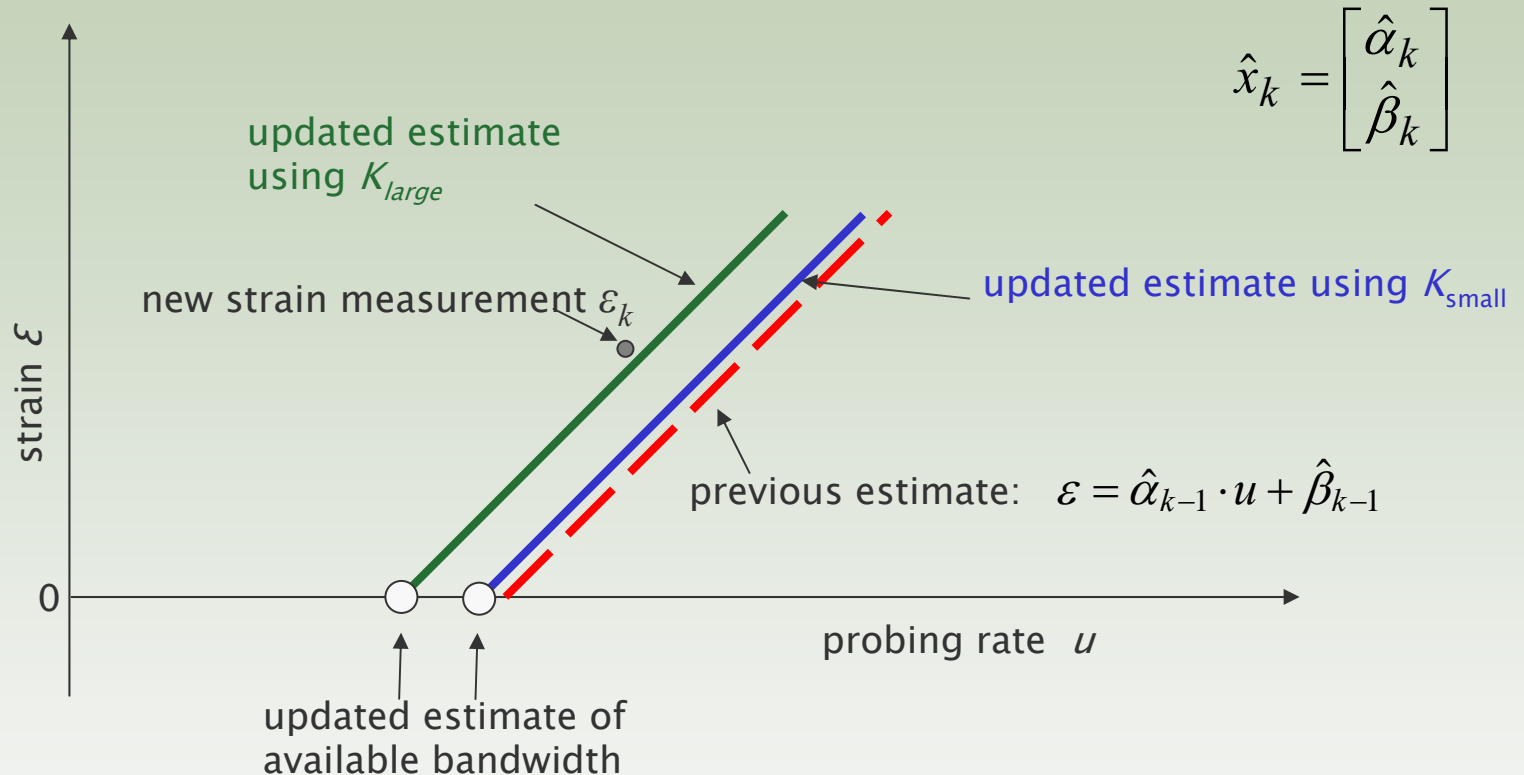
# the heart of the estimator: Kalman filter equations



where the Kalman gain  $K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$   
increases with  $Q$  and decreases with  $R$

# state estimate update (=correction)

- ❖ Kalman gain  $K$  determines how much a new measured  $\mathcal{E}$  affects the system state estimate



- ❖  $Q$  may be used for tuning the filter temporal properties since  $Q$  affects the Kalman gain  $K$

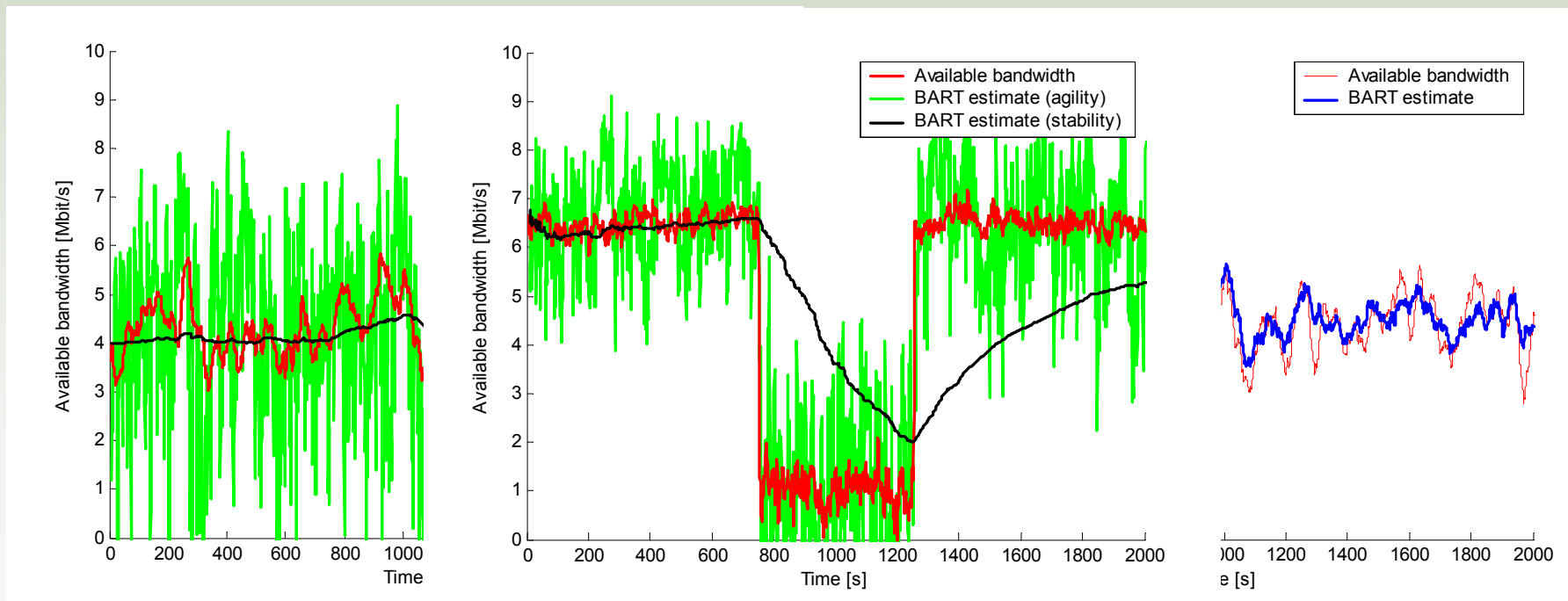
$$Q = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} = C(\Delta x) = \begin{bmatrix} V(\Delta\alpha) & C(\Delta\alpha, \Delta\beta) \\ C(\Delta\alpha, \Delta\beta) & V(\Delta\beta) \end{bmatrix}$$

- ❖ typically for fixed network,  $Q_{22}$  is the most important element

# tuning temporal characteristics

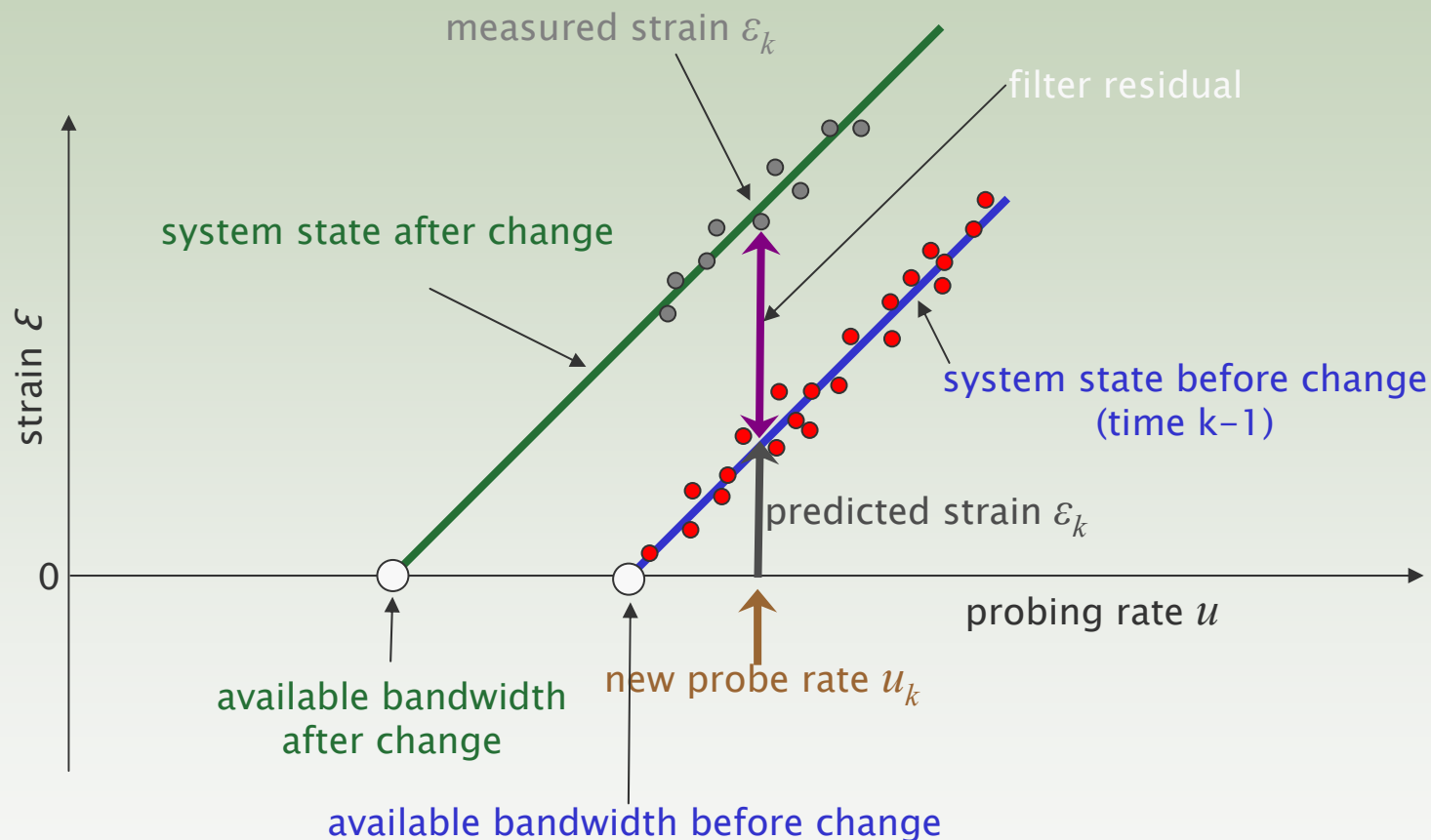
## ❖ Kalman gain $K$ affects BART's temporal tracking characteristics

- BART tracking is more agile if  $K$  is large (i.e.  $Q$  is large)
- BART tracking is more stable if  $K$  is small (i.e.  $Q$  is small)



# change detection

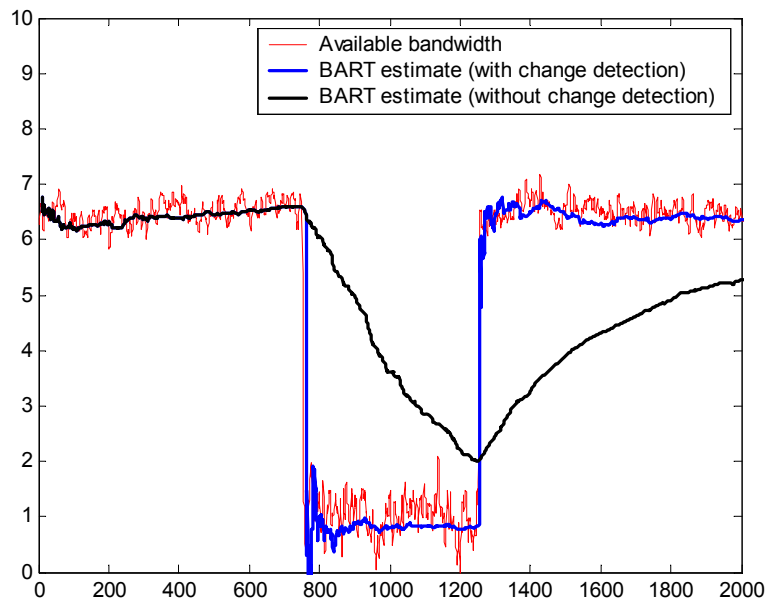
## ❖ filter residuals illustrated:



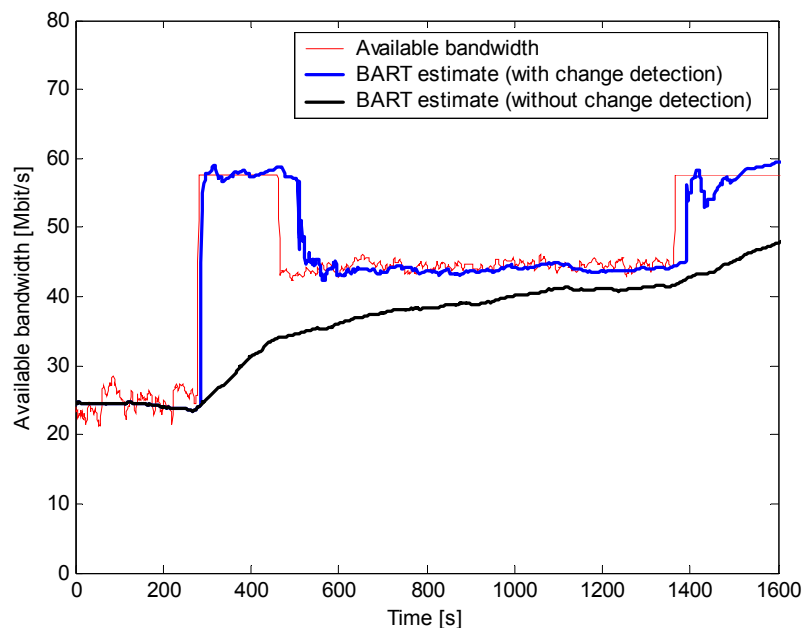
# change detection

- ❖ using a change detection algorithm enables both stable estimation in normal case and quick adaptation to sudden changes of system state

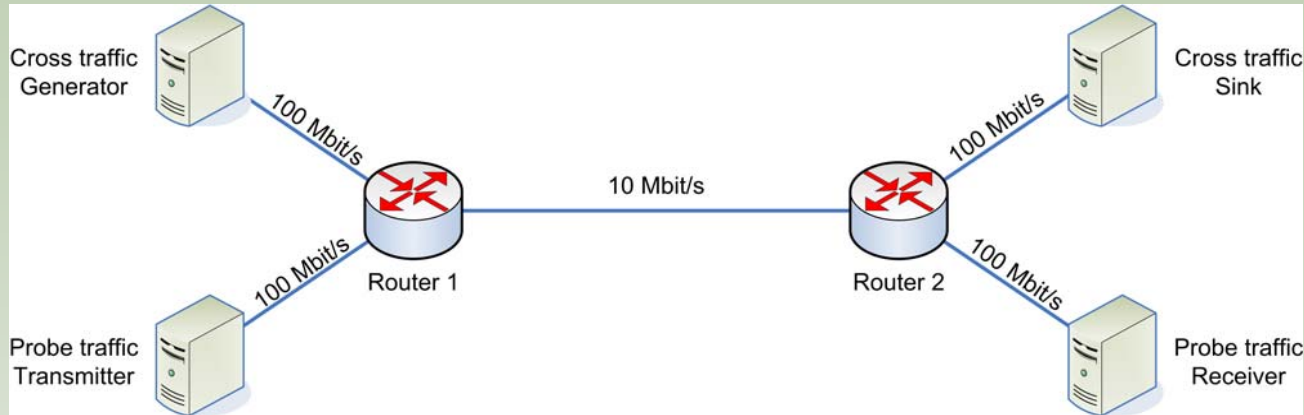
testbed:



Internet:

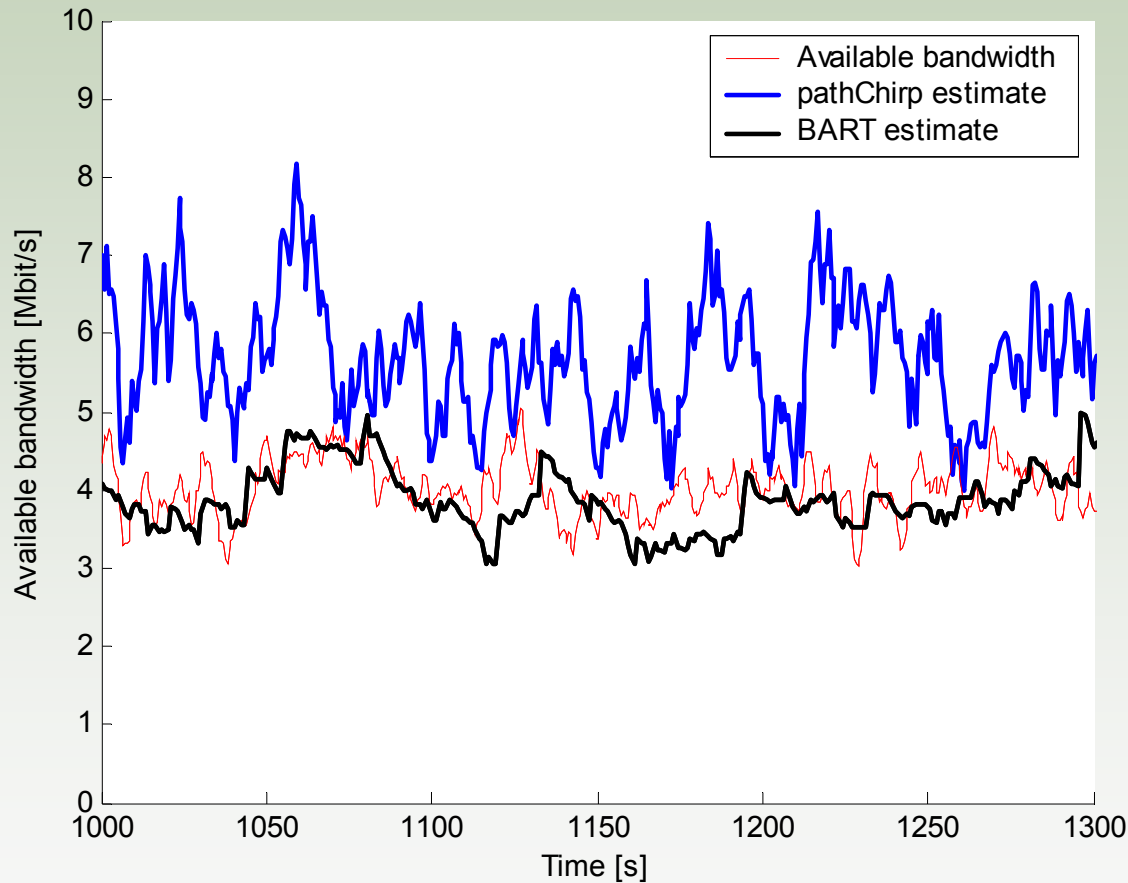


# empirical validation



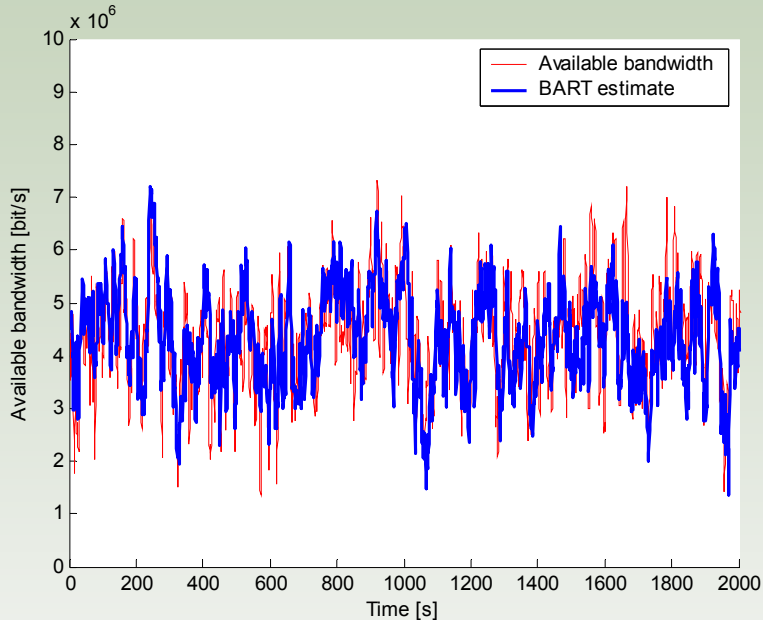
- ❖ **A BART implementation was tested in a lab network**
  - 16 probe packet pairs organized as a train of 17 packets (1500B)
  - one sample per second, uniformly chosen between 1 and 20 Mbit/s
    - 0.2 Mbit/s probe traffic overhead on average
- ❖ **Various traffic cases were emulated using a packet generator**
  - high/low user traffic aggregation
  - different distributions for inter-arrival time
    - exponential
    - Pareto
- ❖ **True available bandwidth was recorded using `tcpdump` for reference**

# Typical results

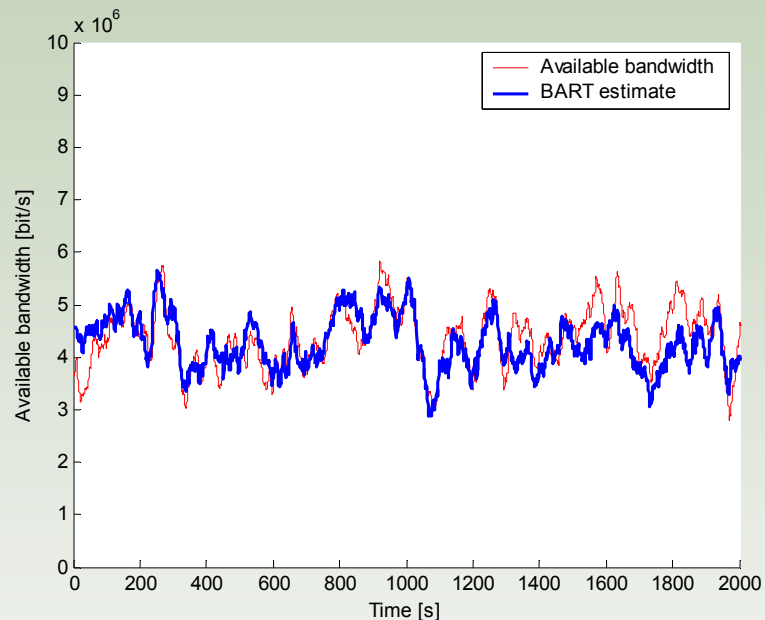


Aggregated traffic, ~100 users, Pareto characteristics

# Tuning BART to desired time scale



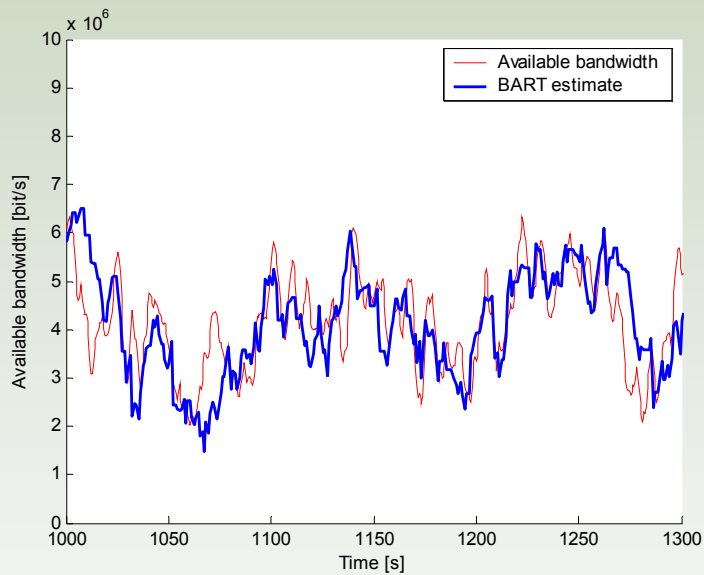
Averaging time scale: 4 s



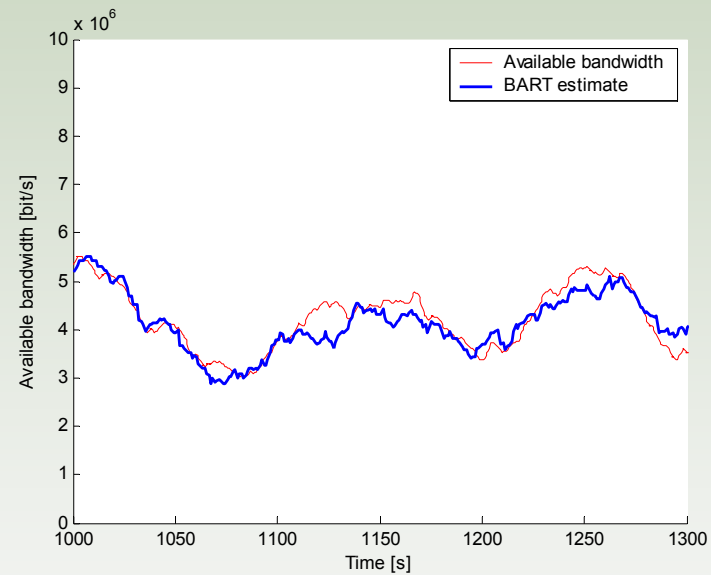
Averaging time scale: 32 s

Note that the true traffic is the same in both graphs.  
(Aggregated traffic,  $\sim 10$  users, Pareto characteristics)

# zooming in...



Averaging time scale: 4 s

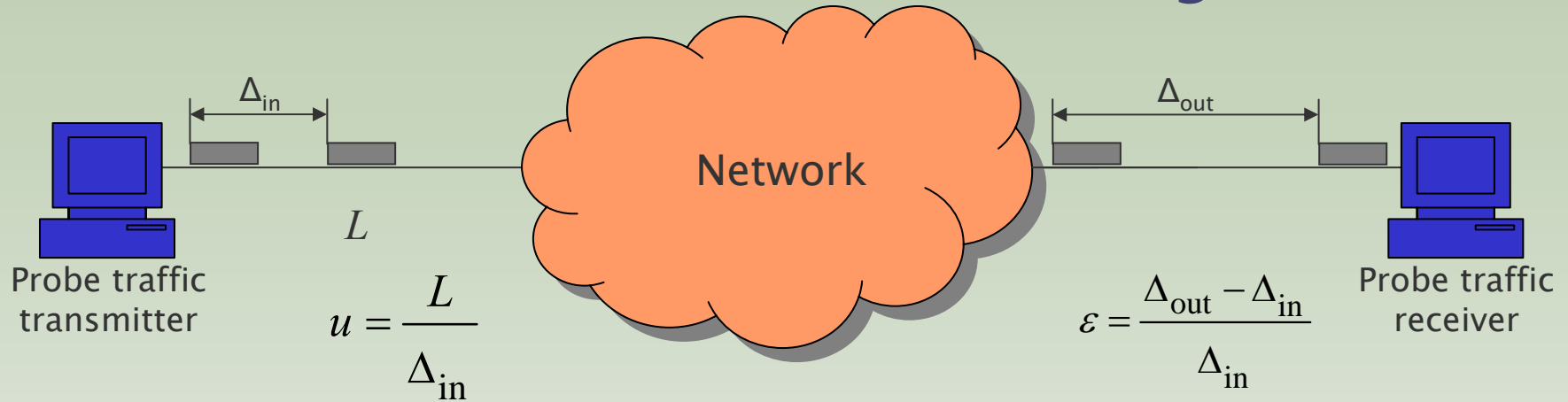


Averaging time scale: 32 s

# BART modeling summary

- ❖ **follows the spirit of Occam's Razor:**
  - avoid making more assumptions than needed
  - a model should be as simple as possible, but no simpler
- ❖ **describes the system state by the two most critical quantities**
  - bottleneck available bandwidth
  - bottleneck capacity
    - (or rather, a transformation of these, to obtain linearity)
- ❖ **formulates the models in a linear fashion**
  - enables Kalman filtering

# BART execution summary



- ❖ samples the path with a sequence of probe-packet pairs (or train) at a rate  $u$
- ❖ randomizes  $u$  for each sampling
- ❖ receiver computes the average strain, inputs this to the Kalman filter, which produces an updated estimate

# BART

- ❖ **does not presume prior knowledge of path**
  - topology
  - characteristics (capacity etc)
- ❖ **needs only access to end hosts – no support from intermediate nodes required**
- ❖ **effective algorithm**
  - > lightweight w r t CPU and memory req.
- ❖ **updates bandwidth estimate for each probe packet sequence**
  - > real-time properties
- ❖ **tunable as to the tracking time-scale**
- ❖ **adaptable to sudden changes of the system state**
- ❖ **patent owned by Ericsson**

# Thank you!

svante.ekelin@ericsson.com