CISTER - Research Center in
Real-Time & Embedded Computing Systems

Information Processing for Extreme Dense Sensing

Eduardo Tovar
CISTER Research Centre
School of Engineering (ISEP) of the Polytechnic of Porto (IPP)
Porto, Portugal
e-Mail: emt@isep.ipp.pt
URL: http://www.dei.isep.ipp.pt/~emt

Instituto Superior de Engenharia do Porto

INESCTEC

This work is made

possible thanks to:

Björn Andersson, Nuno Pereira,
Filipe Pacheco, Maryam Vehabi,
Michele Albano,
Raghuraman Rangarajan,
Vikram Gupta, João Loureiro

Motivation:

Dense Sensing

Welcome

Dense Sensing

- Moore’s law
  - cost (and size) of a single embedded computer node with sensing, processing and (wireless) communication capabilities drops towards zero
    - economically feasible to deploy very large and dense computer networks of such nodes
      - to take very large number of sensor readings from the physical world
      - to compute quantities and take decisions out of those sensor readings
    - the trend is to connect embedded computers through communication networks in order to collaboratively infer and control the state of the physical processes

Dense Sensing

- large scale, dense sensor deployments
  - can cover a large area
  - can offer a better resolution
    - higher quality of sensing/control (e.g., capability of detecting the occurrence of an event)
  - but typically, applications are not interested in all sensor readings, but in computing a function based on sensor readings
    - e.g., MIN or AVERAGE
    - more complex functions
      - e.g., finding the most likely location of an object based on sensor readings

Dense Sensing

- but, these networked embedded computers are
  - resource-constrained
    - typically battery-operated
    - with reduced computing and communication capabilities
    - therefore energy-efficient operation is important
  - and, because of the physical interaction, it is often necessary that the delay from sensing until actuation (decision) is low and bounded

Dense Sensing

- … the challenge is then how
  - to perform <u>scalable and efficient information processing</u> in such large-scale, dense cyber-physical systems
    - with:
      - (i) low delay
      - (ii) low resource usage
  - what do we mean by scalability and efficiency?
    - "efficient information processing"
      - the desired computation is performed while consuming very little resources (energy, communication links, memory, processor)
    - "scalable"
      - consumption of resources increases slowly or not at all as the number of sensor readings to be processed and/or the number of embedded computer nodes increases

A Co-Design Approach

Dense Sensin

Co-Design Approach

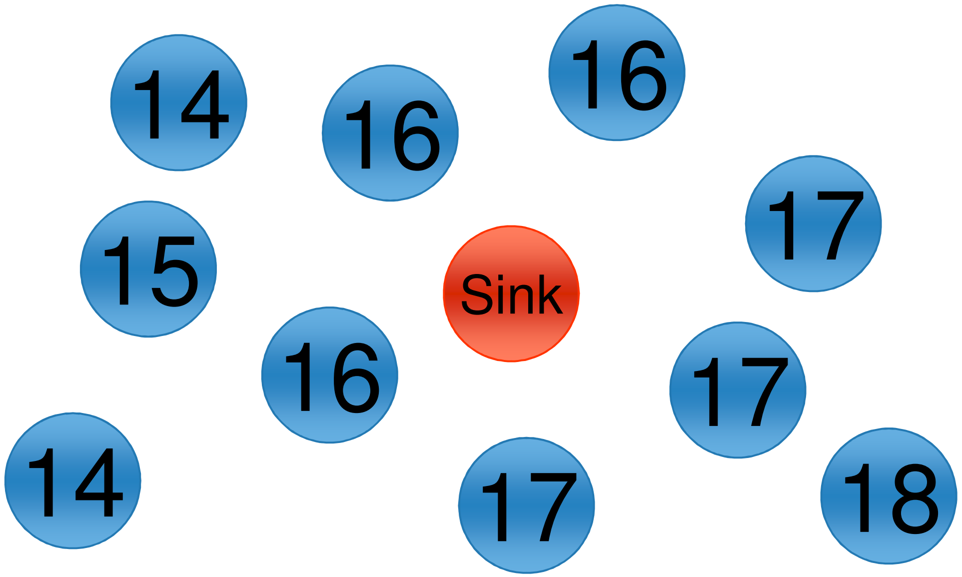
- the problem of performing scalable and efficient information processing in large-scale CPS must be solved
  - otherwise the usefulness of large scale, dense deployments is reduced significantly

- we believe that it is important to take a “clean-slate” approach
  - in order to attain the best possible performance for systems in the long term

Co-Design Approach

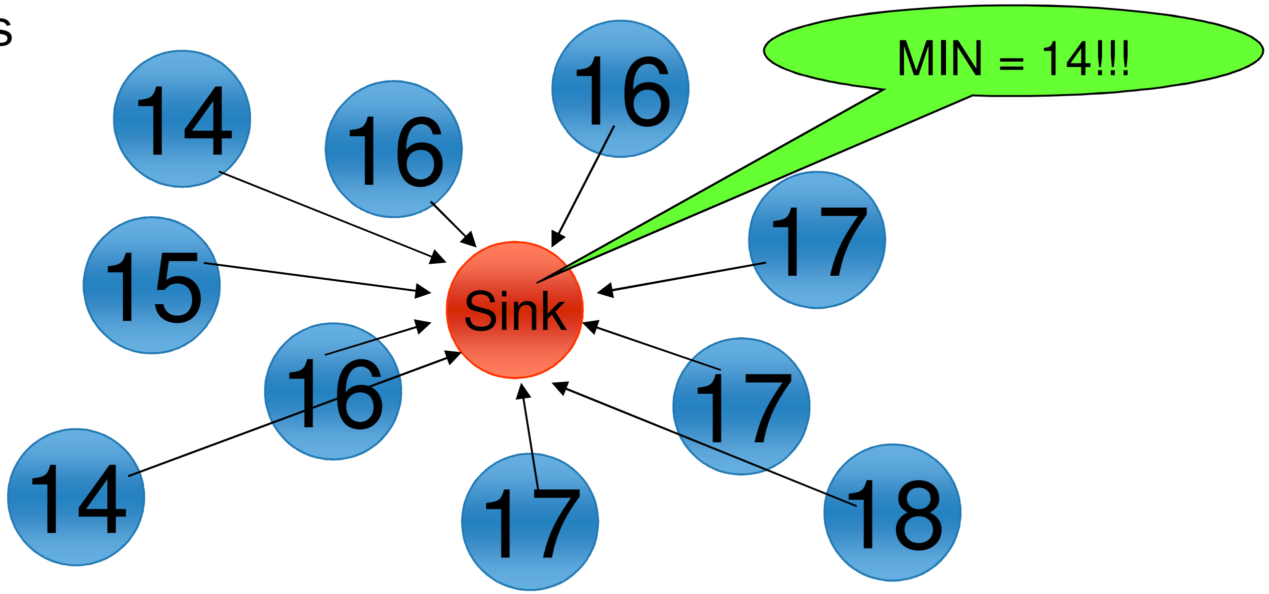
- consider the (simple) problem of computing a simple aggregate quantity such as MIN:
  - the minimum (MIN) sensed temperature (or other physical quantity) among the nodes at a given moment
  - assume the following as being a large and dense deployment
    - 10 nodes (just for the sake of exemplification)

Sorry, it does not seem that large and dense ☹



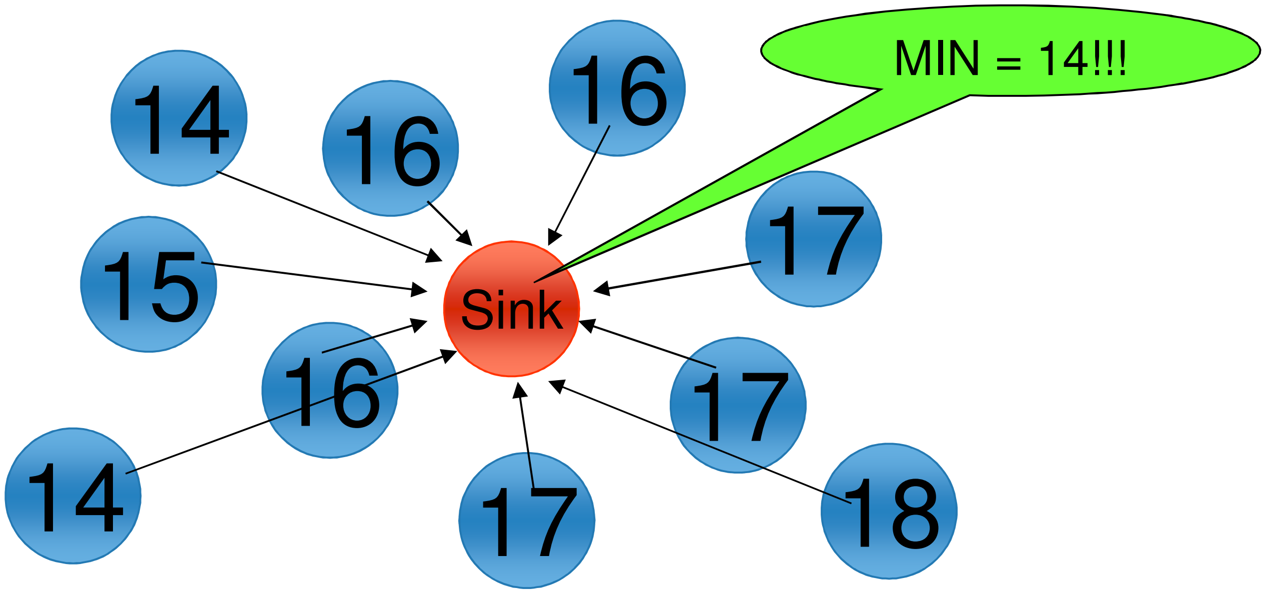
Co-Design Approach

- MIN is trivial, but for systems with large and dense deployment of nodes (such the one in Fig. below 😐)
  - time-complexity as a function of number of nodes (no scalability)
    - this is true even if in-network data aggregation (e.g., convergecast trees) is used
      - since density reduces opportunities for parallel transmission
  - 10 messages



Co-Design Approach

- we have an ambition, though:
  - compute MIN with a time-complexity that is independent of the number of nodes
    - in fact, with a time-complexity that is equivalent to the time of transmitting a single message
      - only possible if all send at the same time...

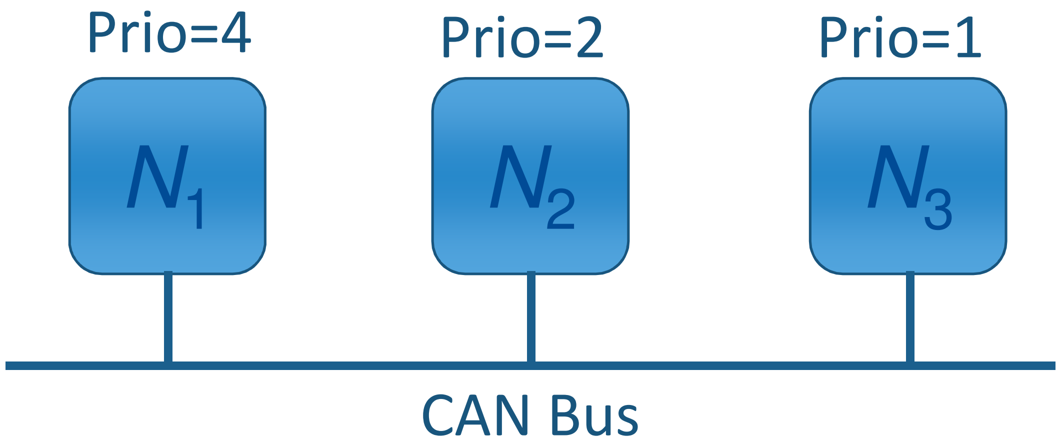


Co-Design Approach

- is such a medium access control (MAC) possible?
  - CAN (Controller Area Network) uses a dominance/binary-countdown protocol (1979 Al Mok's paper "Distributed Broadcast Channel Access", *Computer Networks* 3(2))
    - developed by Robert Bosch GmbH
    - originally for the automotive industry
    - widely used in many other areas (building automation, industrial control, monitoring, ...)
    - millions of nodes and systems deployed
  - characteristics of CAN
    - designed for a wired bus
    - each node (message) has a unique identifier (=priority)
      - lower values for priority mean higher priority
    - resolve bus contention using a bitwise arbitration (non-destructive collision)
      - if a node sends a '1' but hears a '0', he loses
    - notion of recessive and dominant bits
      - 0 is dominant; 1 is recessive
    - bus implements a wired-AND

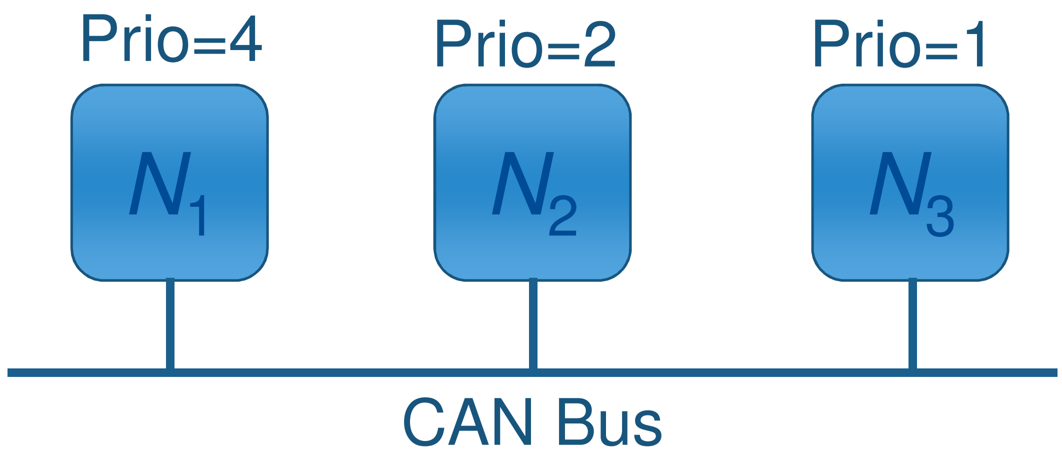
Co-Design Approach

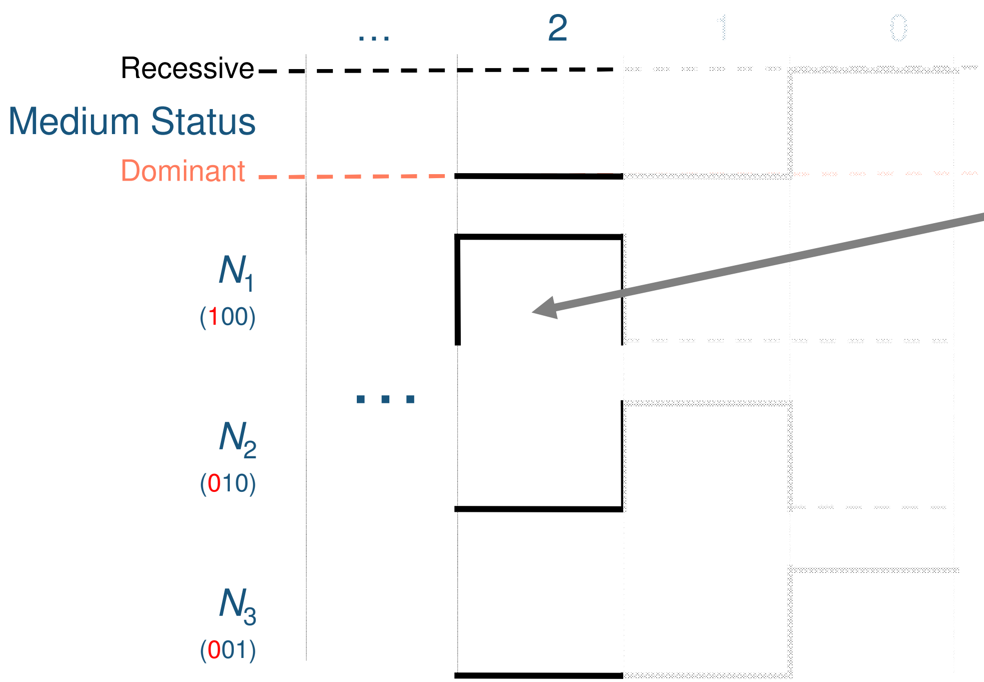
- example CAN network
  - three CAN nodes, with priorities (IDs) 4, 2 and 1



Co-Design Approach

- example (ctnd.)



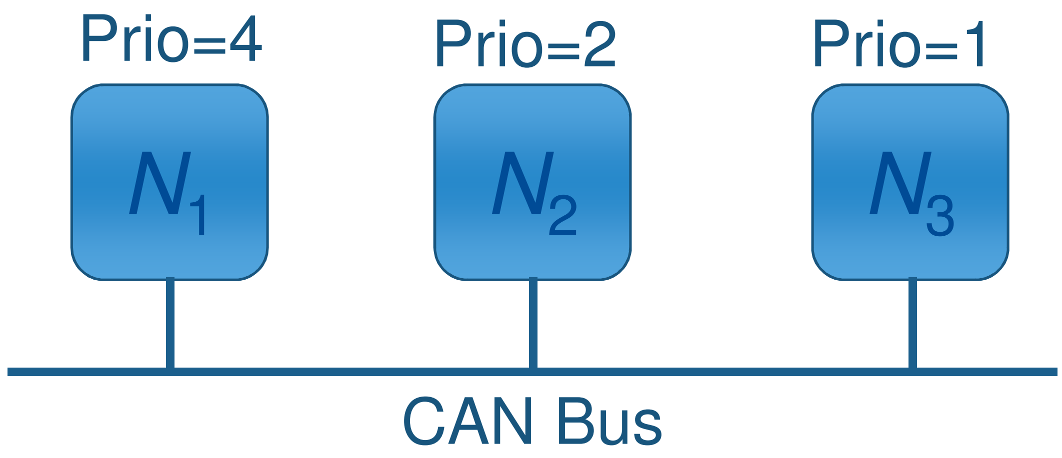


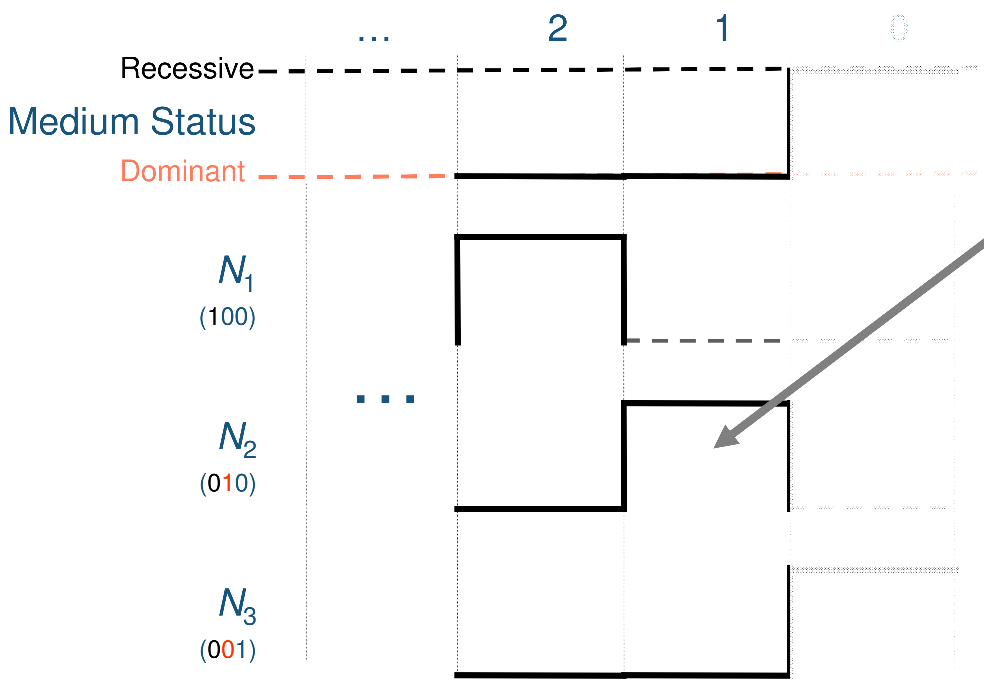
N_1 transmits a recessive bit, but the medium is dominant

N_1 loses the arbitration

Co-Design Approach

- example (ctnd.)



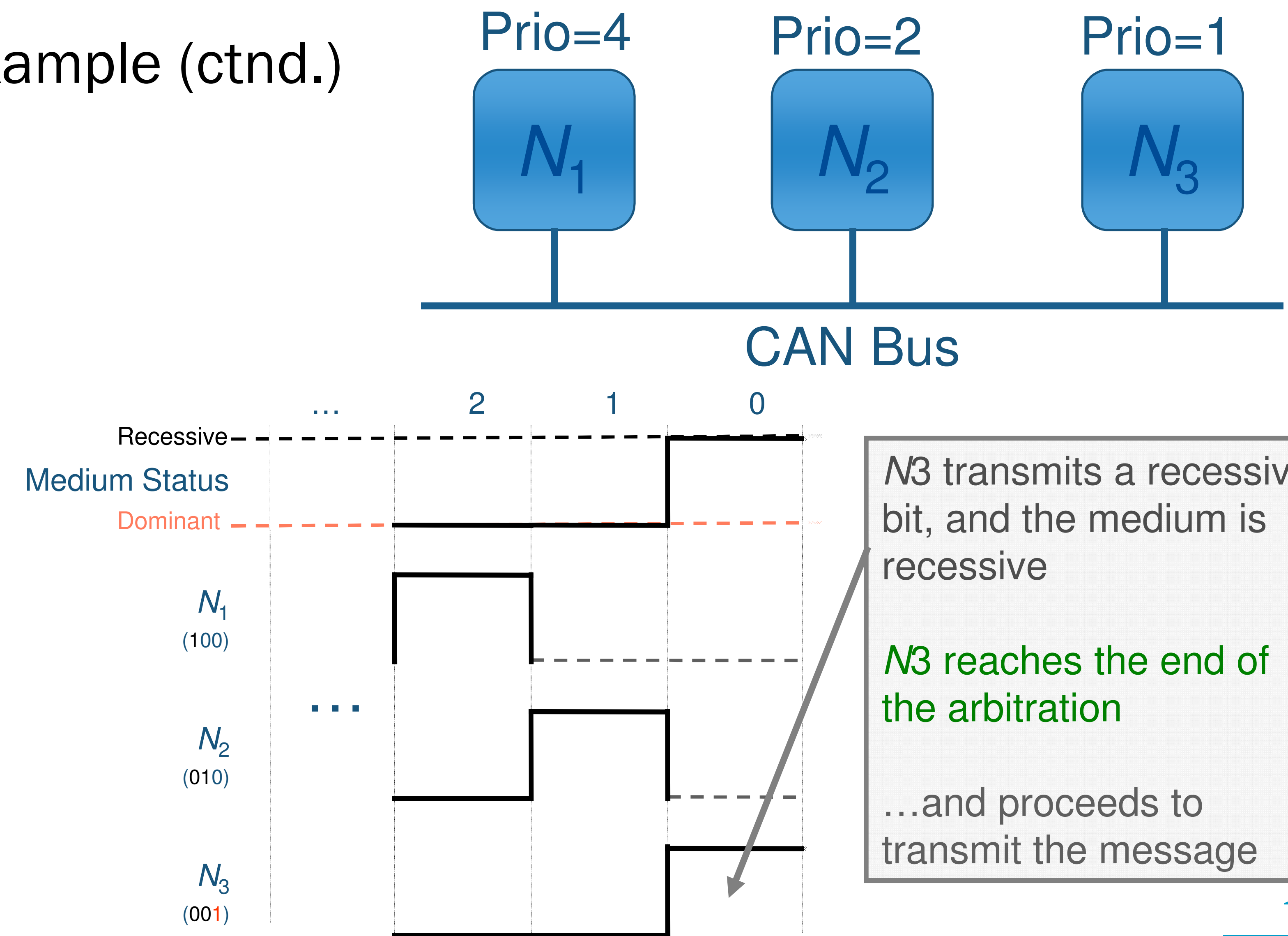


N_2 transmits a recessive bit, but the medium is dominant

N_2 loses the arbitration

Co-Design Approach

- example (ctnd.)



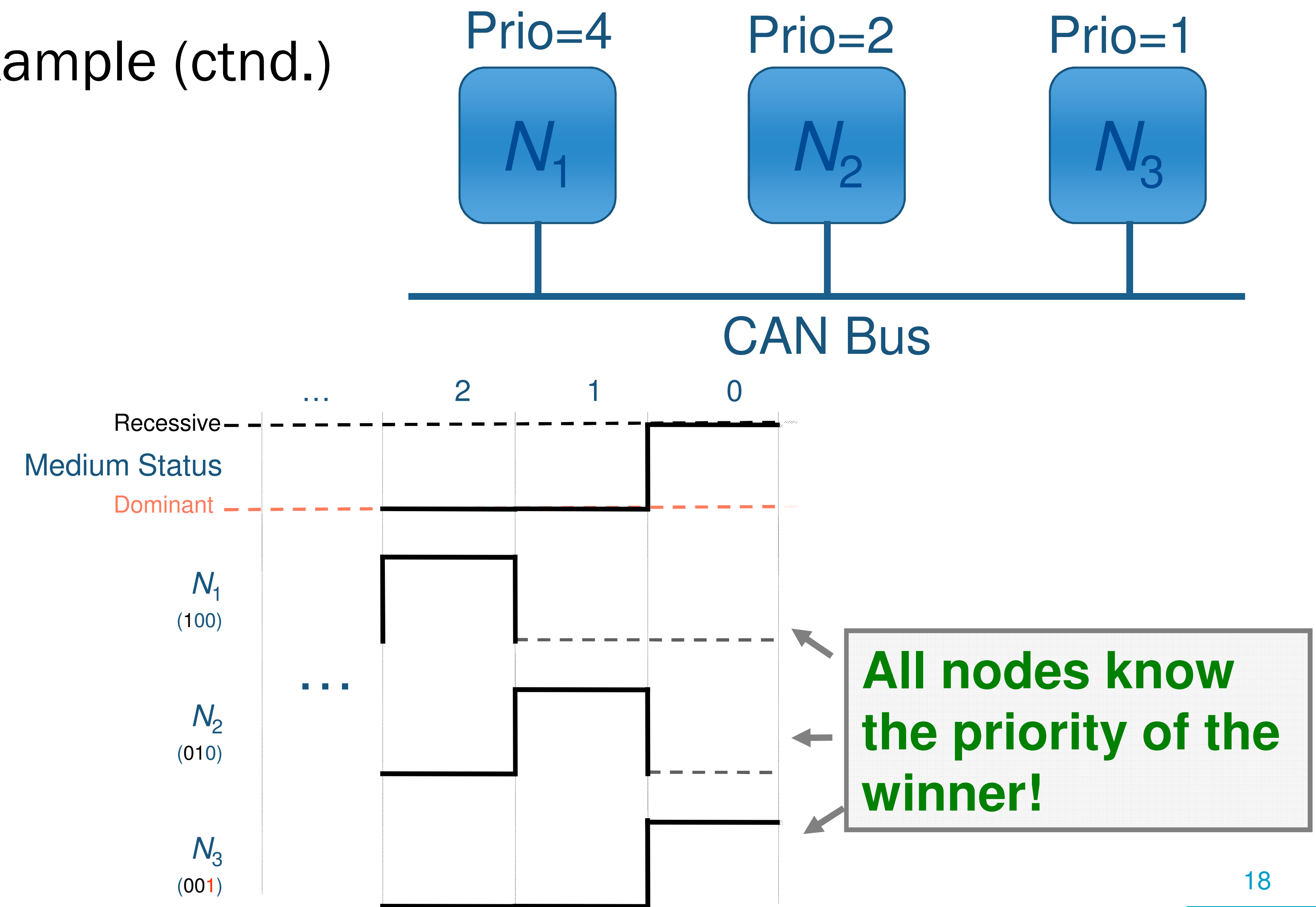
N3 transmits a recessive bit, and the medium is recessive

N3 reaches the end of the arbitration

…and proceeds to transmit the message

Co-Design Approach

- example (ctnd.)



Co-Design Approach

- we propose to use the contention field differently of CAN
  - during runtime, the contention (or priority) field is computed as a function of the physical quantity (or characteristic) of interest
    - it is a Physical Dynamic Priority Dominance ($(PD)^2$) protocol
      - this will be an important building block for computing aggregate quantities with a low time-complexity
      - the $(PD)^2$ protocol is an example where communication and computation is tightly coupled with the physical environment
        - » a clear co-design feature

The End

Q & A

Co-Design Appro

For what the heck is “Min” useful?

Questions & Ans

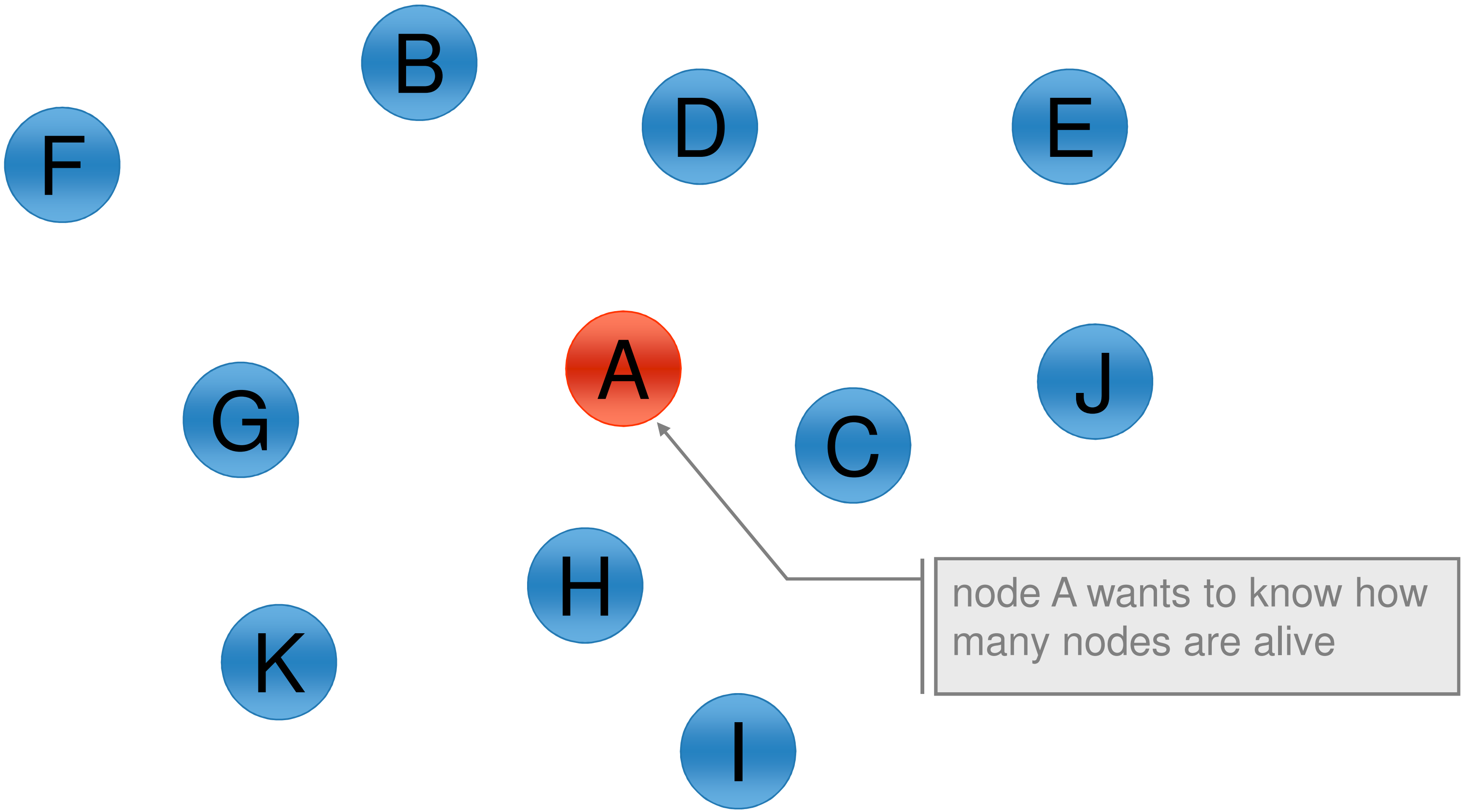
Estimation of COUNT

What the Heck

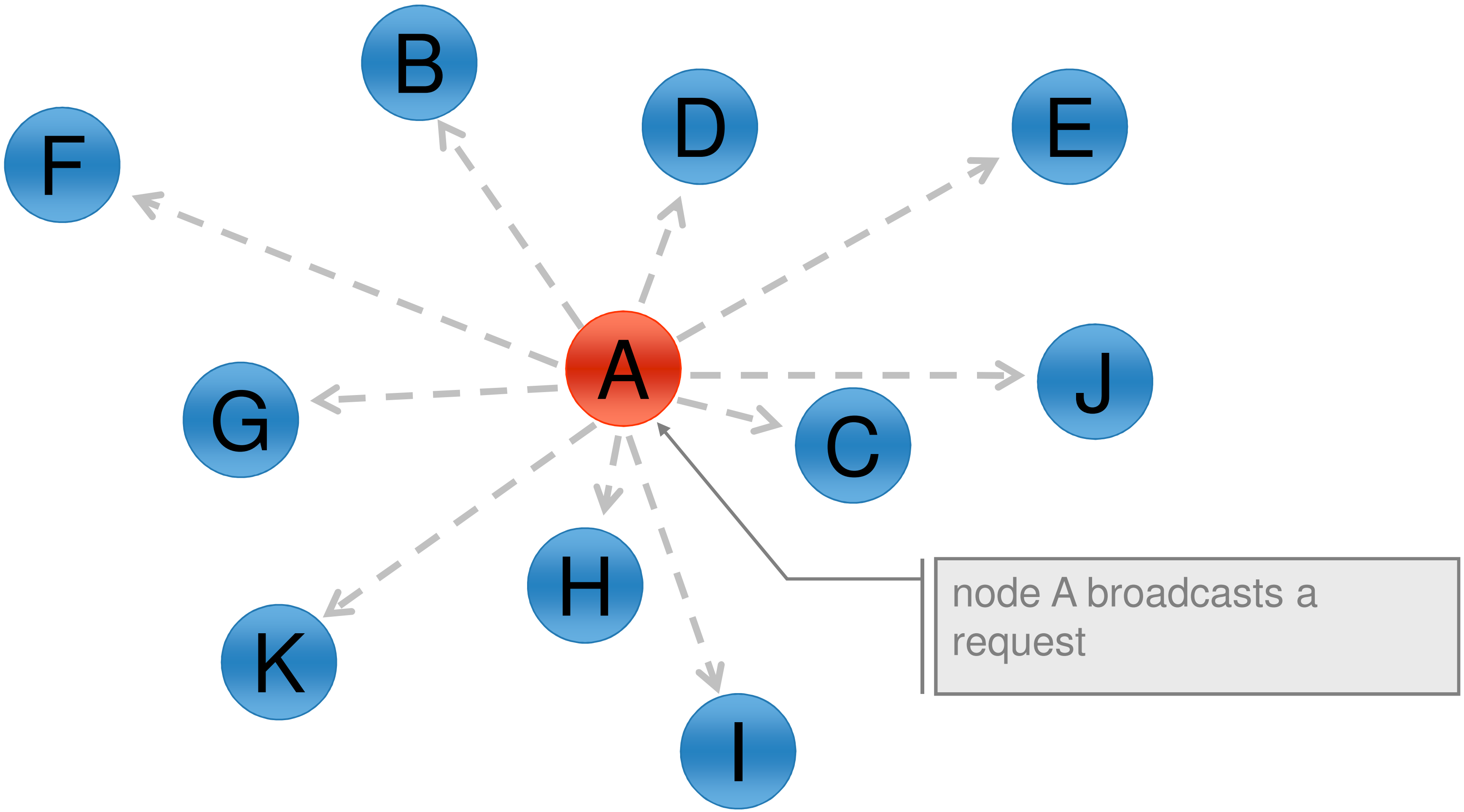
Estimation of COUNT

- intuition behind it
  - if the contention field is a nonnegative random number obtained at runtime, then the probability that the minimum value of the contention field is 0 approaches 1 as the number of nodes get very large
  - however, if there are only a few nodes, then it is highly unlikely that the minimum among the random values is zero
  - it is then possible to estimate the number of nodes by computing the MIN of the random numbers
    - this can with *k* iterations and using Maximum Likelihood estimation
    - MIN is not a function of a sensed physical quantity, instead it is a function of a physical reality

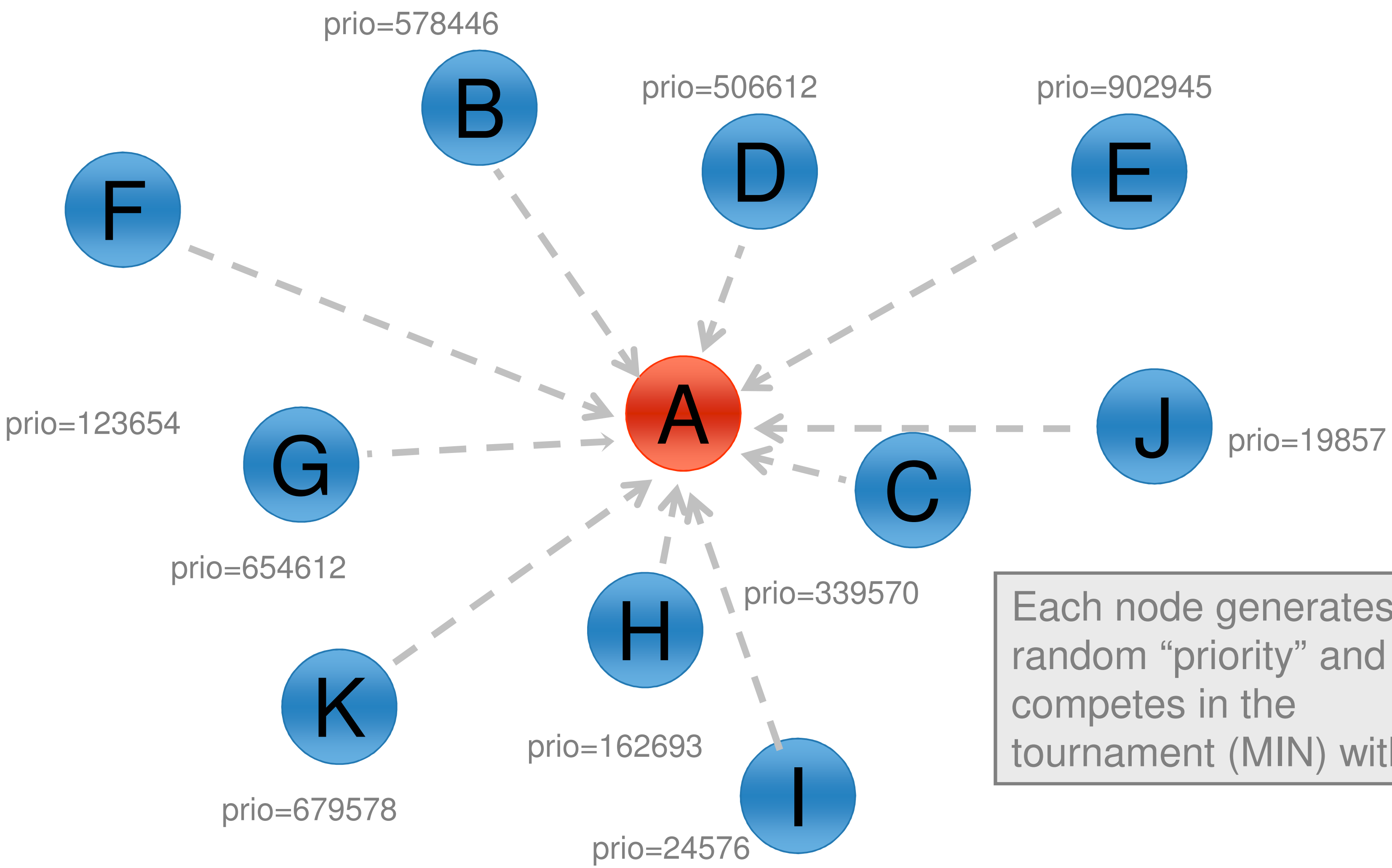
Estimation of COUNT



Estimation of COUNT

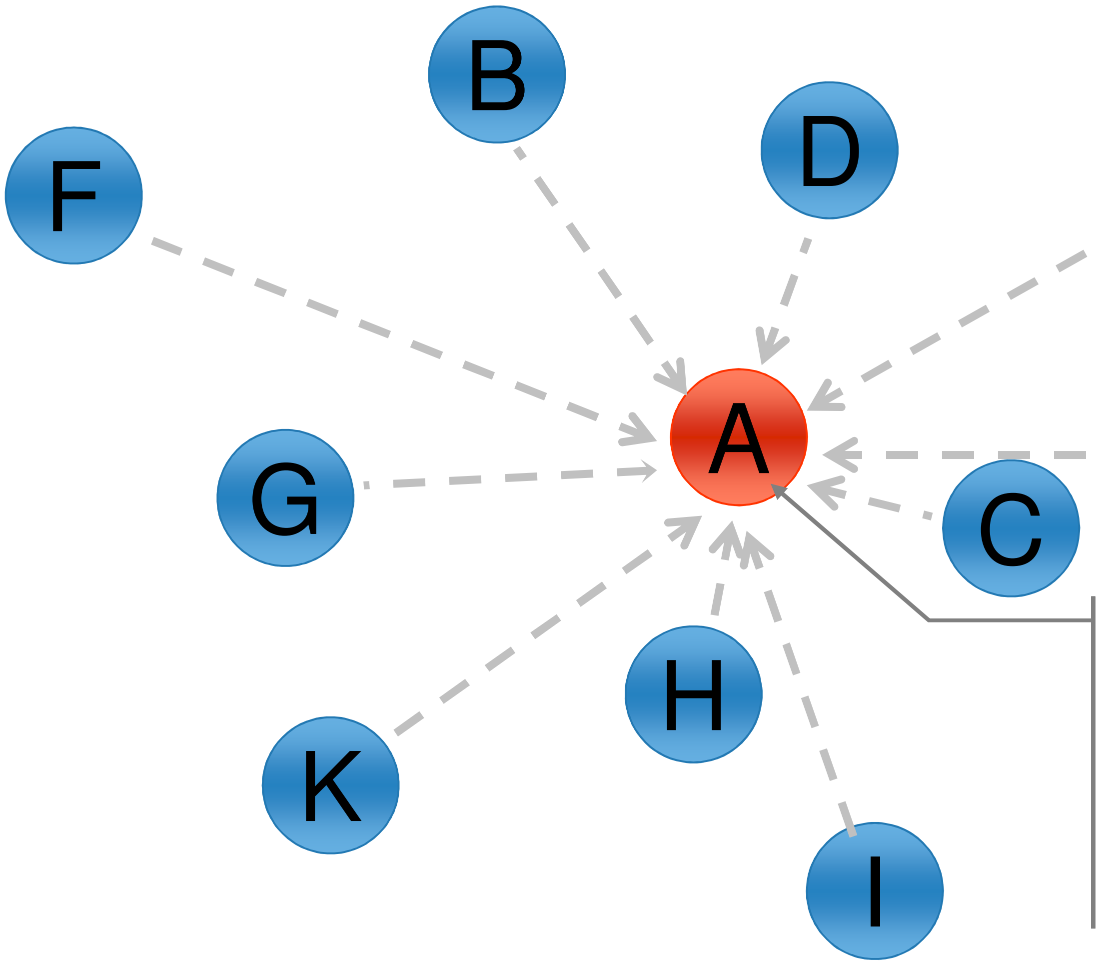


Estimation of COUNT



Each node generates a random "priority" and competes in the tournament (MIN) with it…

Estimation of COUNT



After a single tournament (k=1), node A knows the minimum of the random values, and computes the number of nodes based on a maximum likelihood estimator

Estimation of COUNT

Algorithm 1 Estimating COUNT (the number of nodes)

Require: All nodes start Algorithm 1 simultaneously.
Input: *active* - a global boolean variable indicating if the node is considered in the COUNT

function $nnodes(j : \text{integer}, x : \text{array}[1..k] \text{ of integer})$ **return** a real
$r : \text{array}[1..k]$ of integer
$x : \text{array}[1..k]$ of integer
$q :$ integer
for $q \leftarrow 1$ **to** k
  if $(active = \text{TRUE})$ **then**
    $r[q] \leftarrow random(0, MAXV)$
  else
    $r[q] \leftarrow MAXV$
  end if
  $x[q] \leftarrow$ `send_empty`$(r[q])$
end for
if $(\exists q : x[q] = MAXV)$ **then**
  $est\_nodes \leftarrow 1$
else
  $est\_nodes \leftarrow ML\_estimation(x[1], x[2], ..., x[k])$
end if
return est_nodes *// the estimation of COUNT*

Algorithm 2 Function `ML_estimation`

Require: The division of two integers (as is done in line 6) returns a real number.

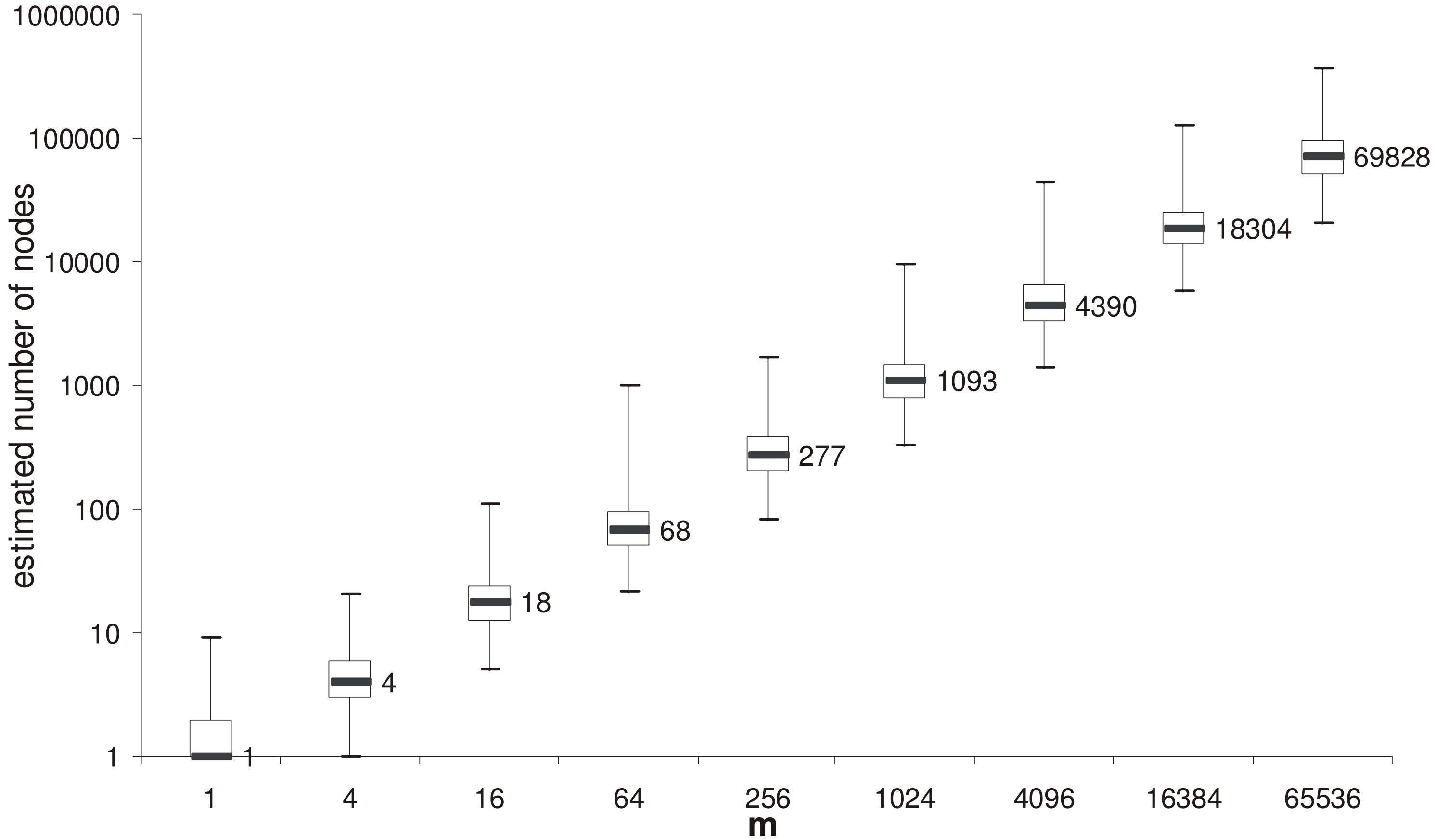
1: **function** $ML\_estimation(x : \text{array}[1..k] \text{ of integer})$ **return** an integer
2: $v : \text{array}[1..k]$ of real
3: $sumv, q :$ integer
4: $sumv \leftarrow 0$
5: **for** $q \leftarrow 1$ **to** k
6: $v[q] \leftarrow ln\left(\frac{1}{1-\frac{x[q]}{MAXV}}\right)$
7: $sumv \leftarrow sumv + v[q]$
8: **end for**
9: **return** $\lceil \frac{k}{sumv} \rceil$
10: **end function**

Estimation of COUNT

– in the two next plots:
  - the box plots in are presented in a logarithmic scale
  - depict the distribution of 1000 estimations for the different numbers of nodes
  - each box stretches from 25th percentile to the 75th percentile
  - the value of the median of the 1000 estimations is depicted as a line across the box
  - the minimum values are depicted below the box and the maximum is above

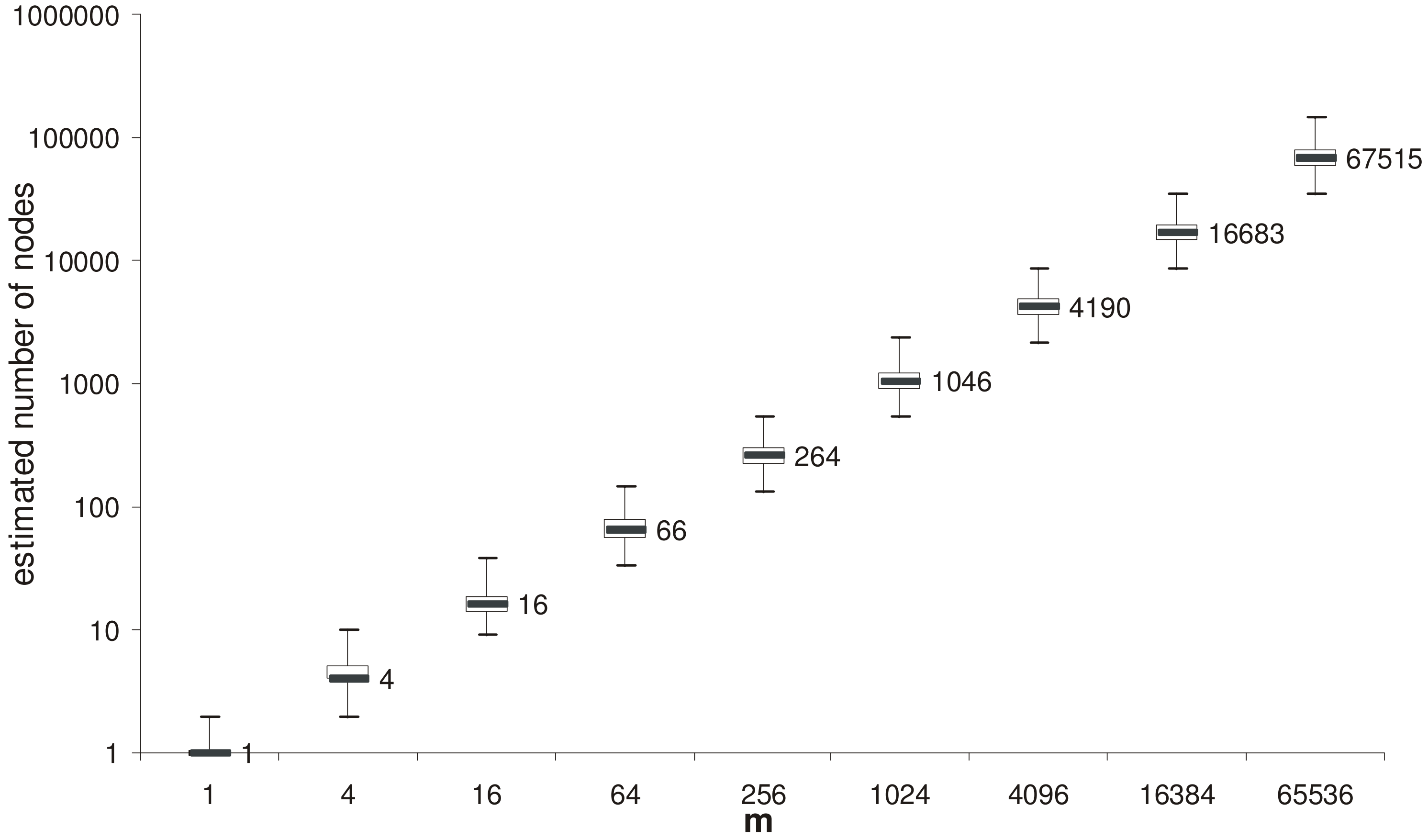
Estimation of COUNT

$k = 5$



Estimation of COUNT

k = 20



Approximate Interpolations

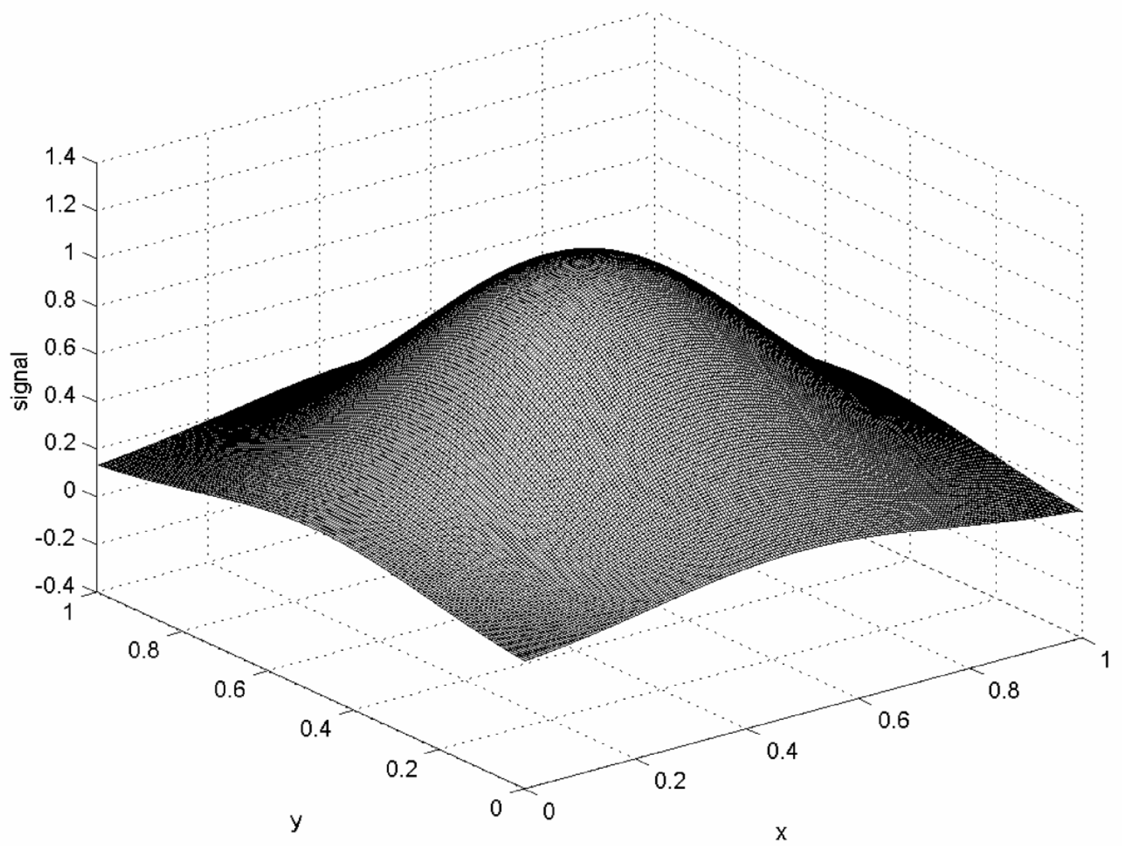
Estimation of COUNT

Approximate Interpolations

– consider a signal (say concentration of a hazardous gas) that varies with location (*x*,*y*)

how can this signal be obtained?

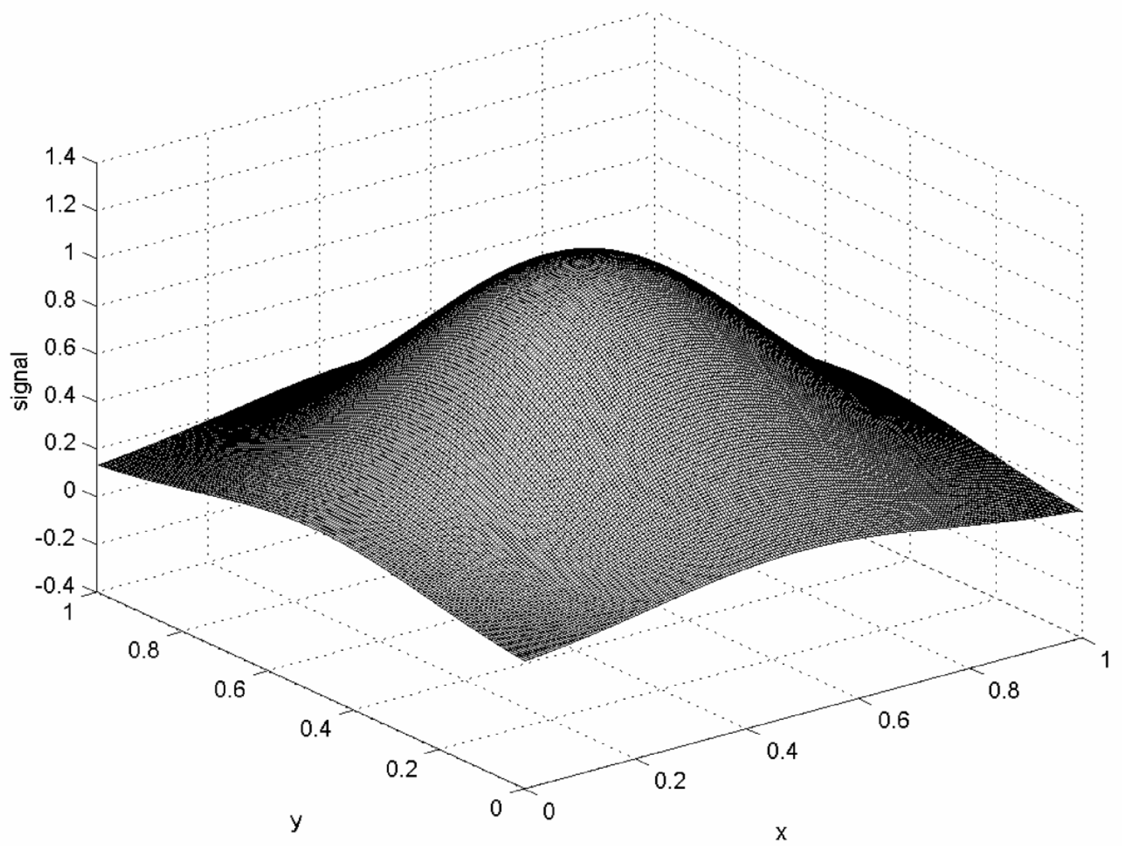
(assume a deployment of a dense network of sensing nodes)



Approximate Interpolations

– consider a signal that varies with location (*x*,*y*)

the signal changes with time… and one wants to get the signal a few times per second…

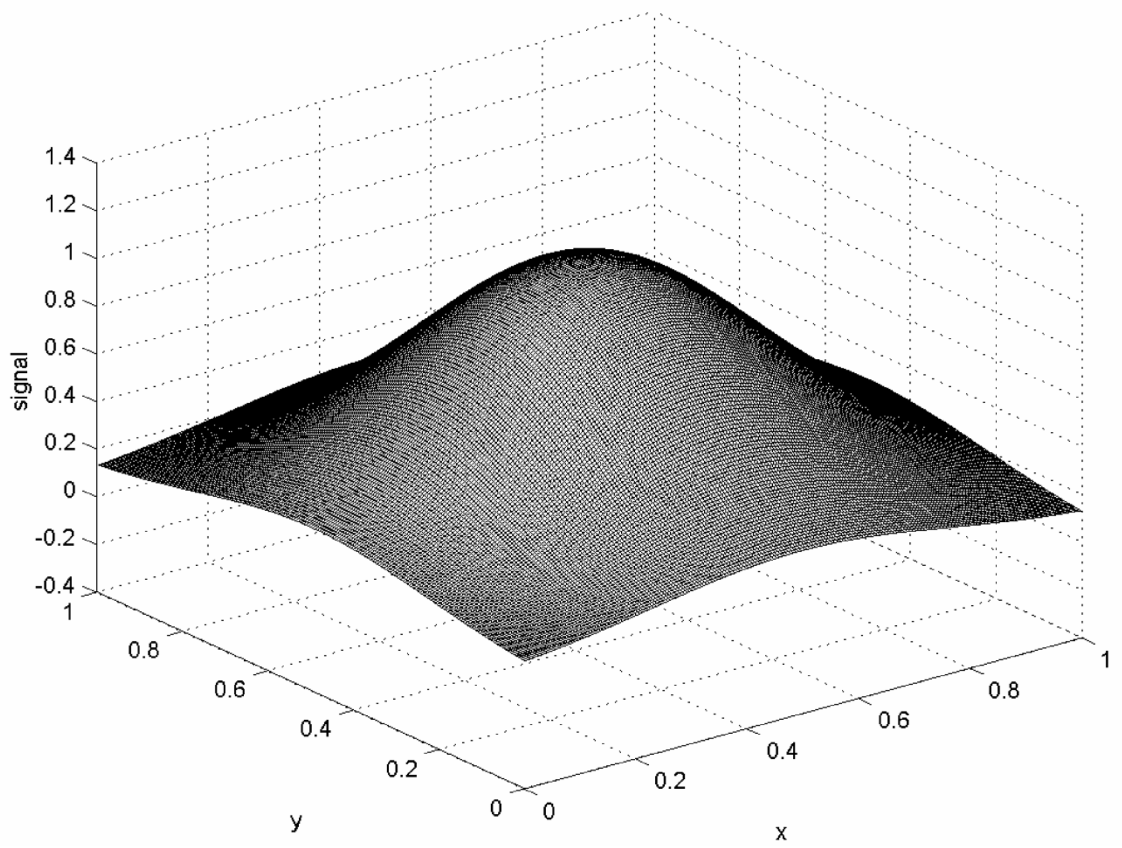


Approximate Interpolations

– consider a signal that varies with location (x,y)

how can this signal be obtained?

one can get sensor readings from **all** the nodes and then perform curve fitting… but it would be slow: O(m)

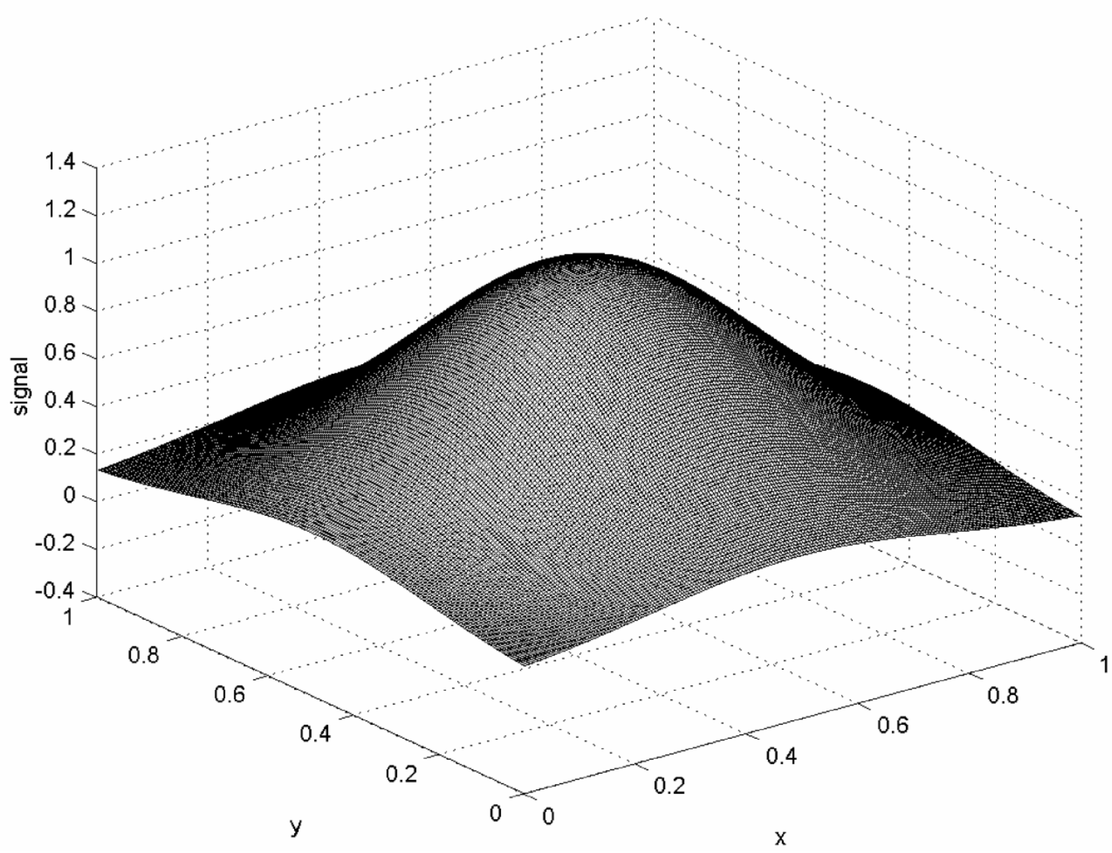


Approximate Interpolations

– consider a signal that varies with location (*x*,*y*)

how can this signal be obtained?

one can get sensor readings from **some of** the nodes and then perform curve fitting… but from which sensor nodes?



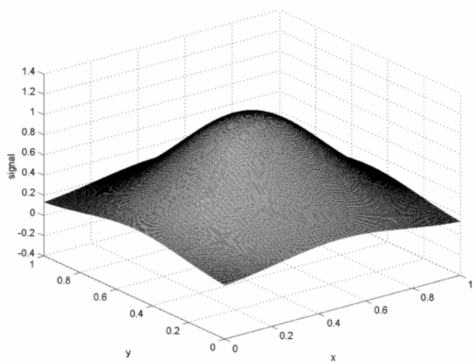
Approximate Interpolations

– consider a signal that varies with location (*x*,*y*)

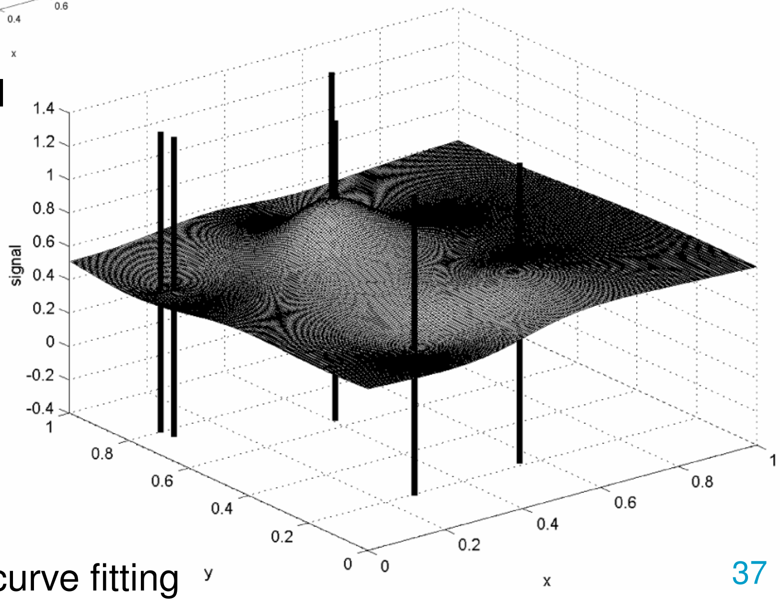
how can this signal be obtained?

one can get sensor readings from **some of** the nodes and then perform curve fitting… but from which sensor nodes?

Select nodes randomly?



original signal



curve fitting

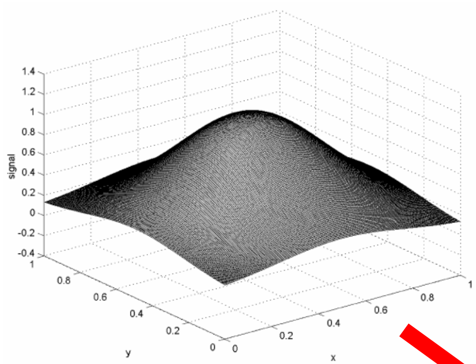
Approximate Interpolations

– consider a signal that varies with location (x,y)

how can this signal be obtained?

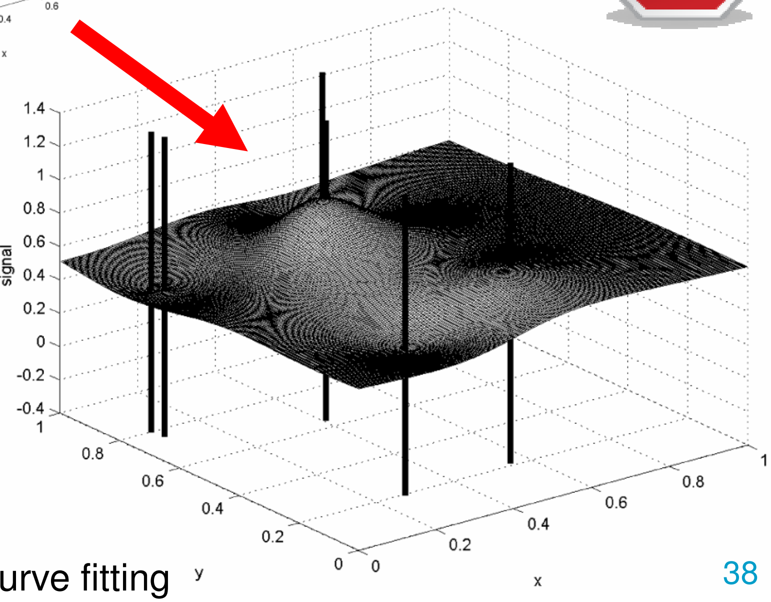
one can get sensor readings from **some of** the nodes and then perform curve fitting… but from which sensor nodes?

Select nodes randomly?



original signal

bad match



curve fitting

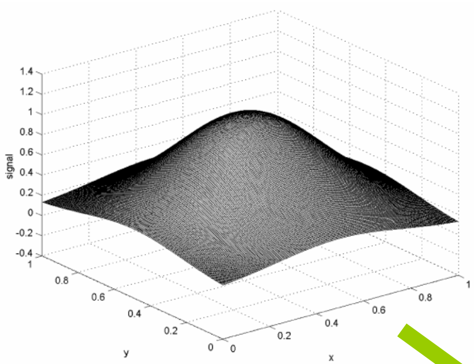
Approximate Interpolations

– consider a signal that varies with location (*x*,*y*)

how can this signal be obtained?

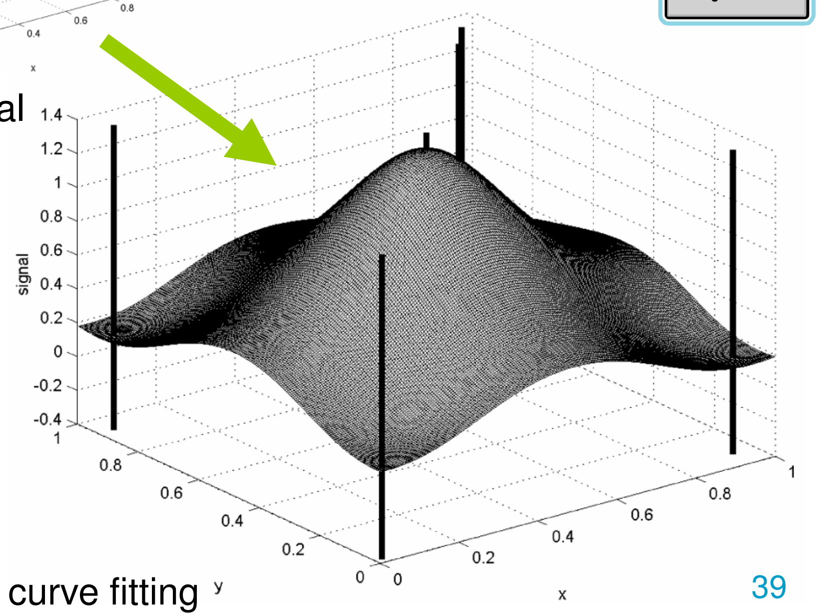
one can get sensor readings from **some of** the nodes and then perform curve fitting… but from which sensor nodes?

Carefully selected nodes



original signal

good match



curve fitting

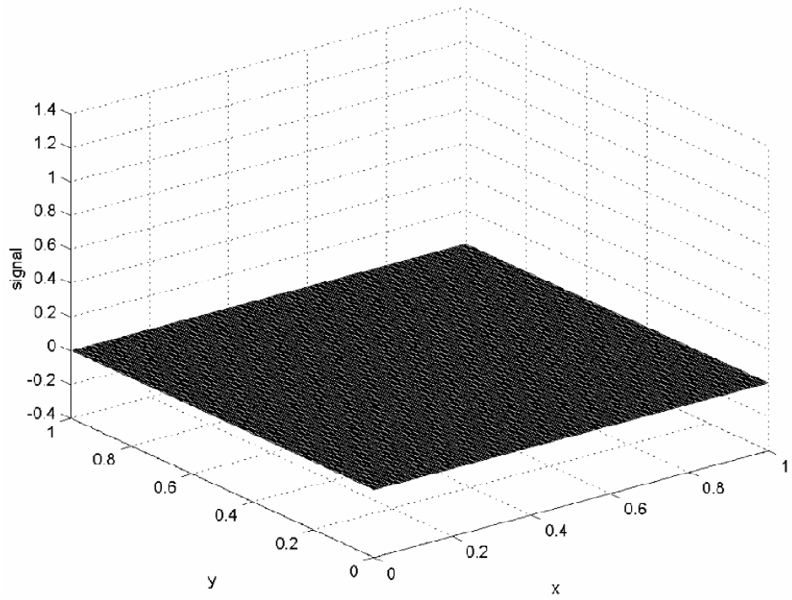
Approximate Interpolations

– consider a signal that varies with location (x,y)

how to carefully select nodes?

iteratively select the node with highest error

start with a flat surface no data points (nodes) selected



curve fitting

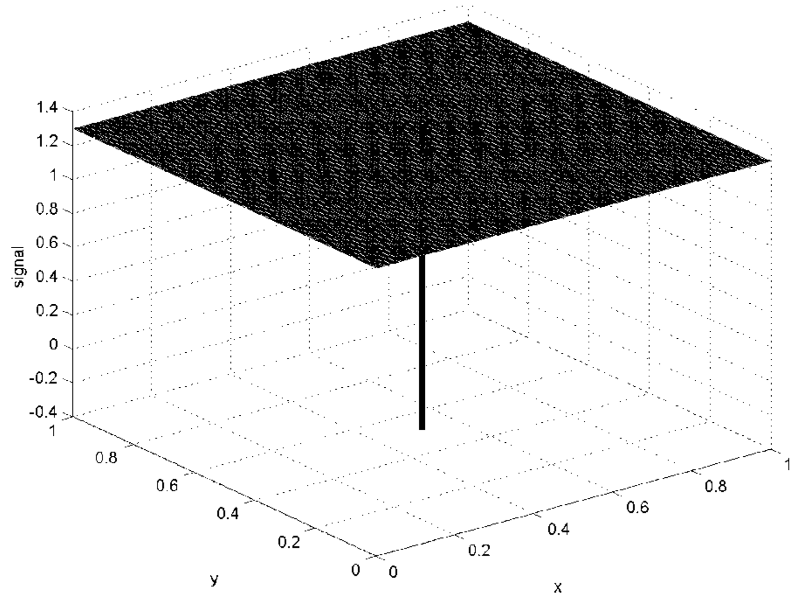
Approximate Interpolations

– consider a signal that varies with location (*x*,*y*)

how to carefully select nodes?

iteratively select the node with highest error

1 data point (node) selected



curve fitting

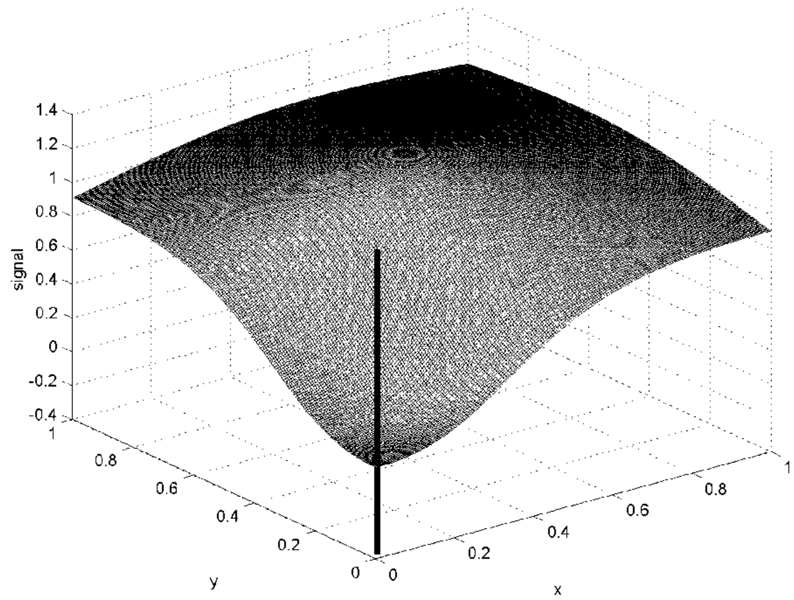
Approximate Interpolations

– consider a signal that varies with location (*x*,*y*)

how to carefully select nodes?

iteratively select the node with highest error

2 data points (nodes) selected



curve fitting

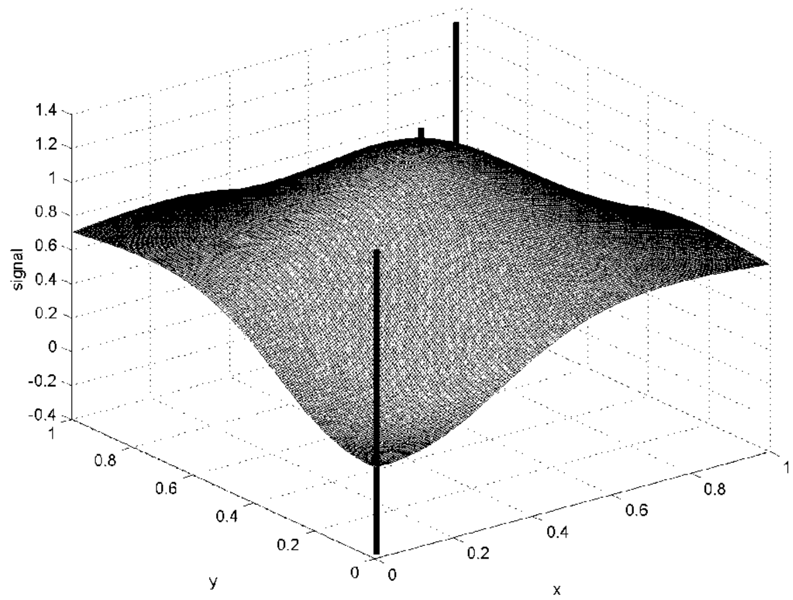
Approximate Interpolations

– consider a signal that varies with location (*x*,*y*)

how to carefully select nodes?

iteratively select the node with highest error

3 data points (nodes) selected



curve fitting

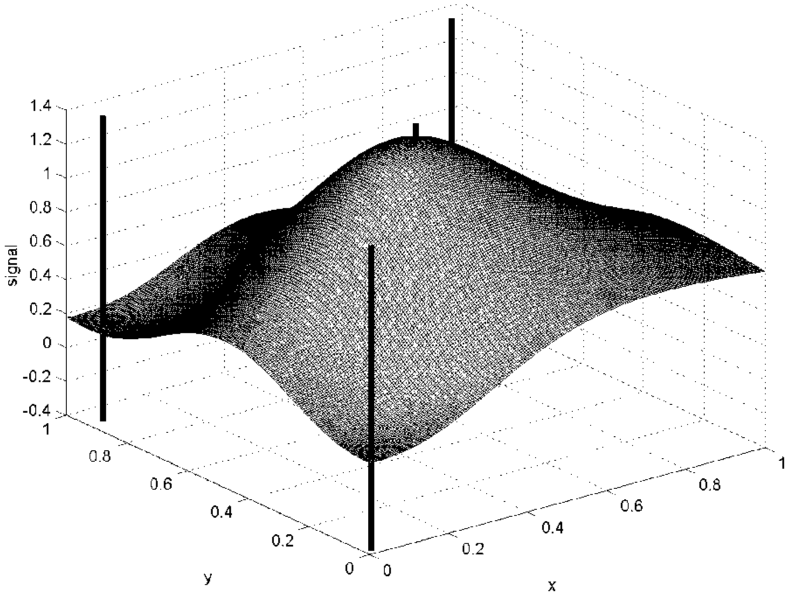
Approximate Interpolations

– consider a signal that varies with location (*x*,*y*)

how to carefully select nodes?

iteratively select the node with highest error

4 data points (nodes) selected



curve fitting

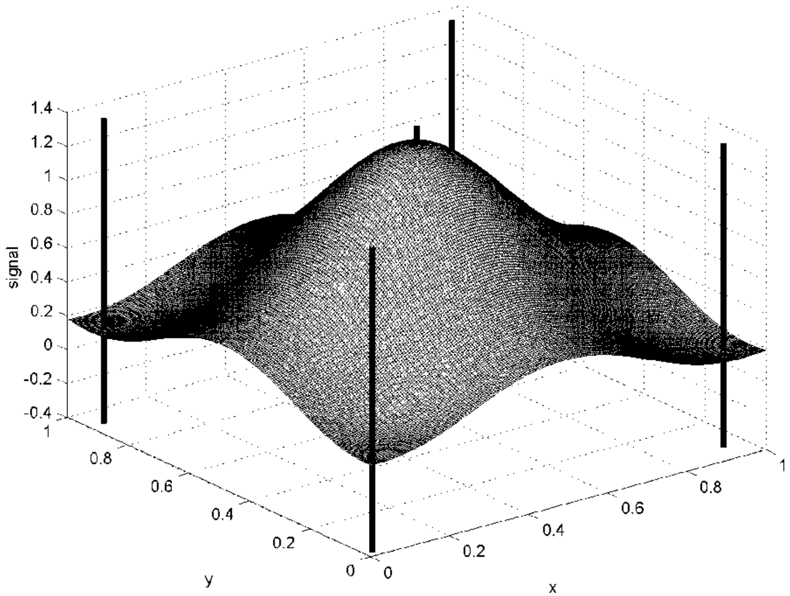
Approximate Interpolations

– consider a signal that varies with location (x,y)

how to carefully select nodes?

iteratively select the node with highest error

5 data points (nodes) selected

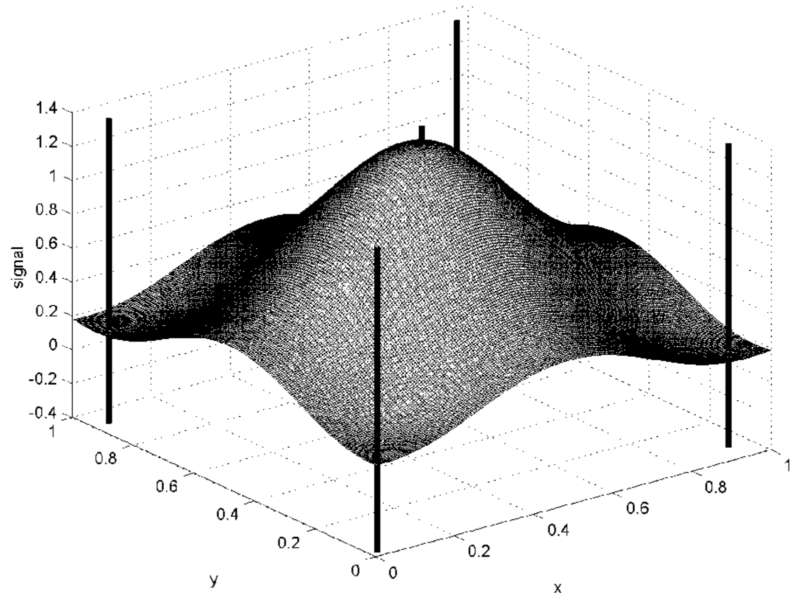


curve fitting

Approximate Interpolations

– consider a signal that varies with location (*x*,*y*)

the (PD)[2] protocol can be exploited to efficiently select the node with priority 1/error!



curve fitting

Approximate Interpolations

– the basic algorithm
  - let *f*(*x*,*y*) denote the function that interpolates the sensor data
  - let e_i denote the magnitude of the error at node N_i; that is:

$$e_i = |v_i - f(x_i, y_i)|$$

  - and let *e* denote the global error; that is:

$$e = \max_{i=1..m} e_i$$

  - the goal is to find *f*(*x*,*y*) that minimizes *e* subject to the following constraint
    – the time required for computing *f* at a specific point is low
      » motivated by the fact that it is interesting to track physical quantities that change quickly; it may be necessary to recompute the interpolation at a relatively high rate
    – therefore, a possibility is to use weighted-average interpolation (WAI) with a selection of relevant points

Approximate Interpolations

– the basic algorithm (ctnd.)

- an approach

$$f(x,y) = \begin{cases} 0 & \text{if } S = \emptyset; \\ v_i & \text{if } \exists N_i \in S \text{ with } x_i{=}x \wedge y_i{=}y; \\ \frac{\sum_{i \in S} v_i \cdot w_i(x,y)}{\sum_{i \in S} w_i(x,y)} & \text{otherwise.} \end{cases}$$

- where *S* is a set of sensor nodes used for interpolation and $w_i(x, y)$ is given by:

$$w_i(x,y) = \frac{1}{(x_i - x)^2 + (y_i - y)^2}$$

- intuitively these equations state that the interpolated value is a weighted sum of all data points in *S* and the weight is the inverse of the square of the distance
  - many possible choices on how the weight should be computed as a function of distance
    - » the way stated above is intended to avoid calculations of square root in order to make the execution time small on platforms that lack hardware support for floating point calculations

Approximate Interpolations

– the basic algorithm (ctnd.)

Algorithm 1 Finding a subset of nodes to be used in WAI

Require: All nodes start Algorithm 1 simultaneously.
Require: k denotes the desired number of interpolation points.
Require: A node N_i knows x_i,y_i and s_i.
Require: MAXNNODES denotes an upper bound on m.
Require: (MAXS+1)$\times$(MAXNNODES+1)+MAXNNODES$\leq$MAXP.

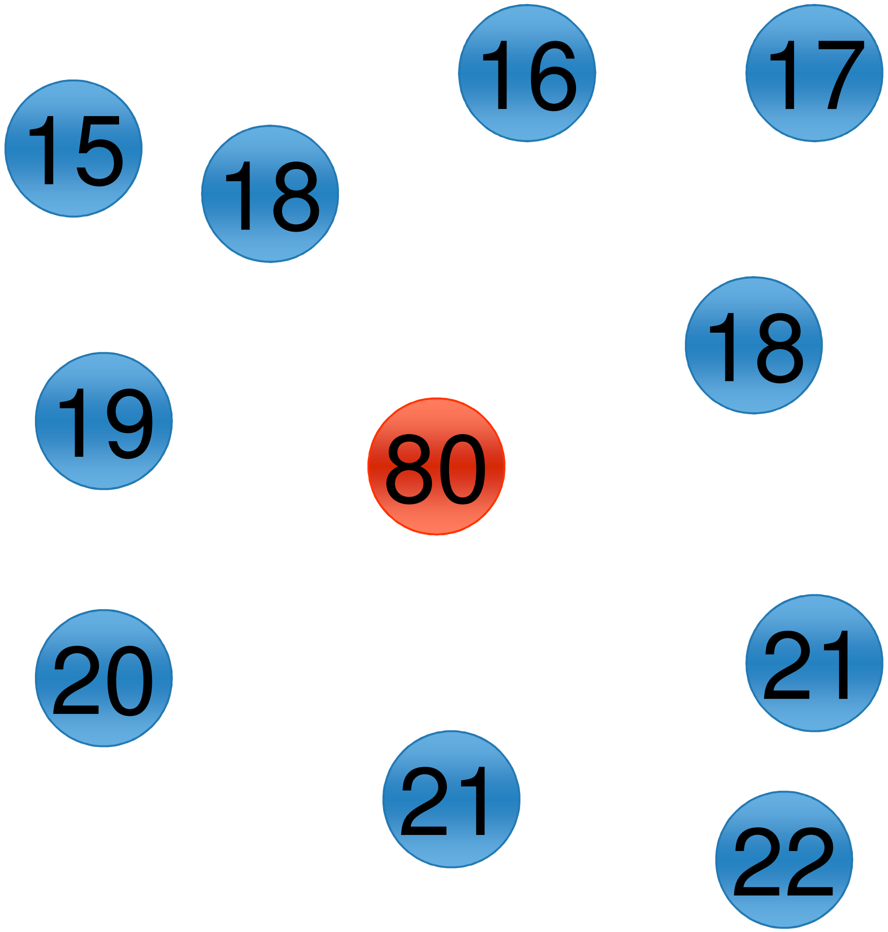
function find_nodes() **return** a set of packets
  myinterpolatedvalue $\leftarrow$ 0
  num $\leftarrow$ 0.0
  denom $\leftarrow$ 0.0
  S $\leftarrow \emptyset$
  update_myinterpolation $\leftarrow$ TRUE
  for j $\leftarrow$ 1 to k **do**
    error $\leftarrow$ abs(s_i - to_integer(myinterpolatedvalue))
    prio $\leftarrow$ MAXP - (error $\times$ (MAXNNODES + 1) + i)
    snd_pack $\leftarrow$ $<s_i,x_i,y_i>$
    <win_prio, rcv_pack> $\leftarrow$ send_and_rcv(prio, snd_pack)
    if win_prio = prio **then**
      update_myinterpolation $\leftarrow$ FALSE
      myinterpolatedvalue $\leftarrow$ s_i
    end if
    if update_myinterpolation = TRUE **then**
      dx $\leftarrow$ x_i - recv_pack.x
      dy $\leftarrow$ y_i - recv_pack.y
      weight $\leftarrow$ 1.0 / (dx $\times$ dx + dy $\times$ dy)
      num $\leftarrow$ num + recv_pack.value $\times$ weight
      denom $\leftarrow$ denom + weight
      myinterpolatedvalue $\leftarrow$ num/denom
    end if
    S $\leftarrow$ S $\bigcup$ recv_pack
  end for
  return S
end function

Approximate Interpolations

– consider a signal that varies with location (*x*,*y*)

tolerance to faulty values?

a single faulty value may jeopardize the whole approach



Approximate Interpolations

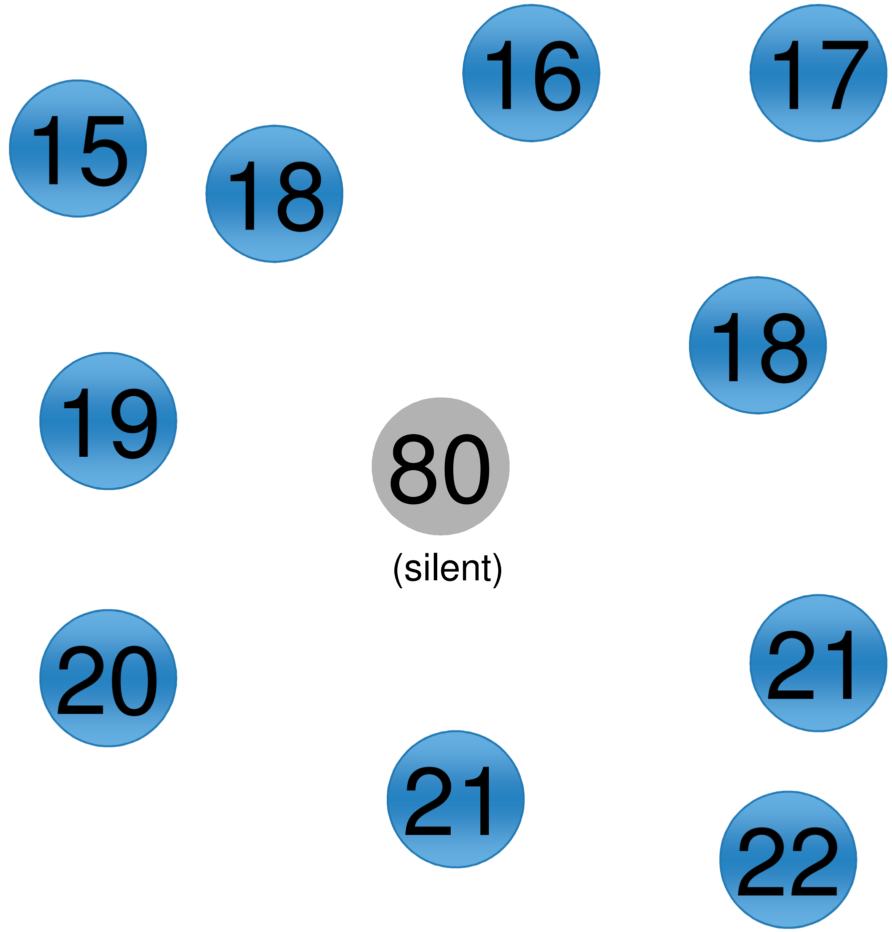
– consider a signal that varies with location (x,y)

tolerance to faulty values?

sensor readings are assumed to exhibit spatial locality

and

correct sensor readings are always in majority



Approximate Interpolations

- as previously (algorithm 1), but then each pair of nodes that were selected in algorithm 1 is inspected and the difference between sensor readings relative to the distance is computed
  - if the value is greater than what is possible by the physical dynamics (a designer parameter) both nodes are declared SILENT
    - this is acceptable since we consider dense networks

Approximate Interpolations

Algorithm 2 Finding a subset of nodes to be used in WAI. This algorithm tolerates sensor faults.

Require: The same requirements as in Algorithm 1

All nodes run Algorithm 1
for each $N_j \in S$ **do**
    $SILENT_j \leftarrow$ FALSE
end if
Let T denote an ordered set containing the elements in S
    Any order is fine as long as T is the same on all nodes.
    Note that S is the same on all nodes.
for each $N_j \in T$ **do**
    for each $N_k \in T$ where N_j is earlier than N_k in T **do**
        sqrds := $(s_j - s_k)^2$
        sqrdist := $(x_j - x_k)^2 + (y_j - y_k)^2$
        if $SILENT_j$ =FALSE **and** $SILENT_k$ =FALSE **and**
            sqrds $>$ THRESHOLD $\cdot$ sqrdist **then**
            $SILENT_j \leftarrow$ TRUE
            $SILENT_k \leftarrow$ TRUE
        end if
    end for
end for
All nodes N_j with $SILENT_j$=FALSE run Algorithm 1

The End?

Q & A

Approximate Interpolation

What kind of applications need this density?!?

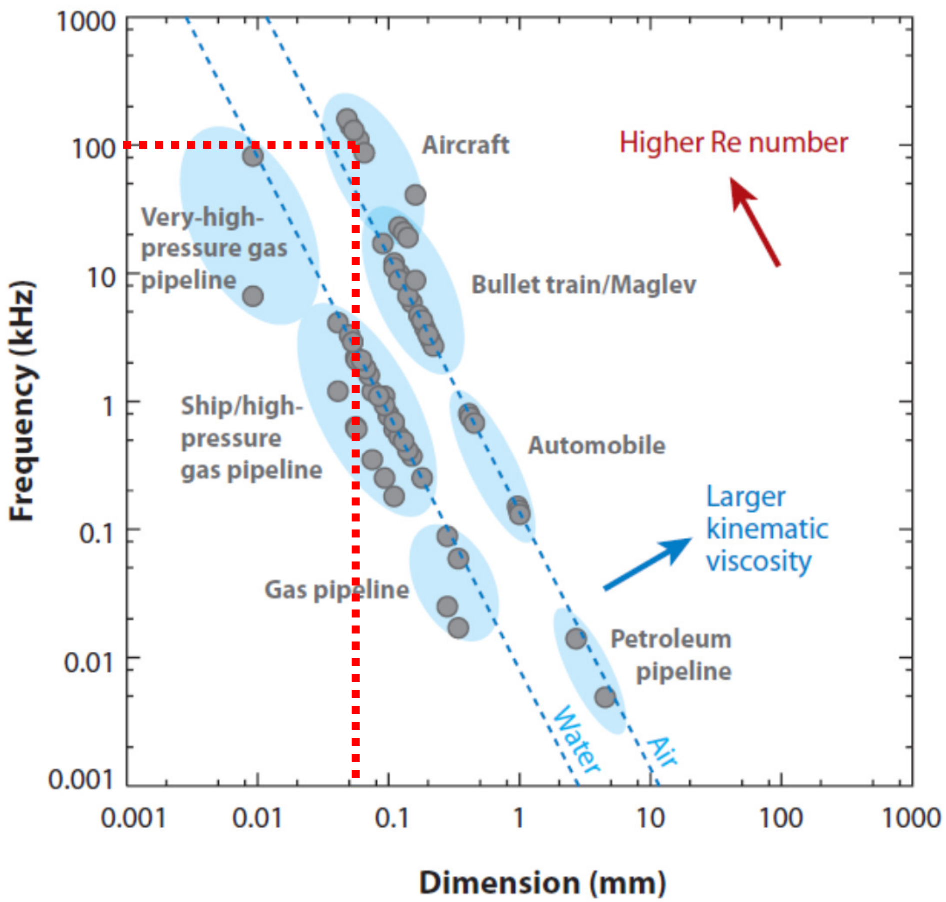
Questions & Ans

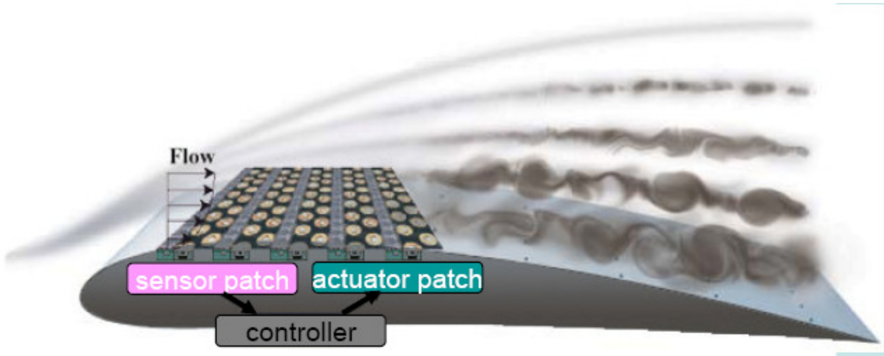
Active Flow Control

What Applicatio

Active Flow Control

– The spatial and temporal scales of the sensors have to be small enough compared to the turbulent structures. (100kHz and 0.1mm)



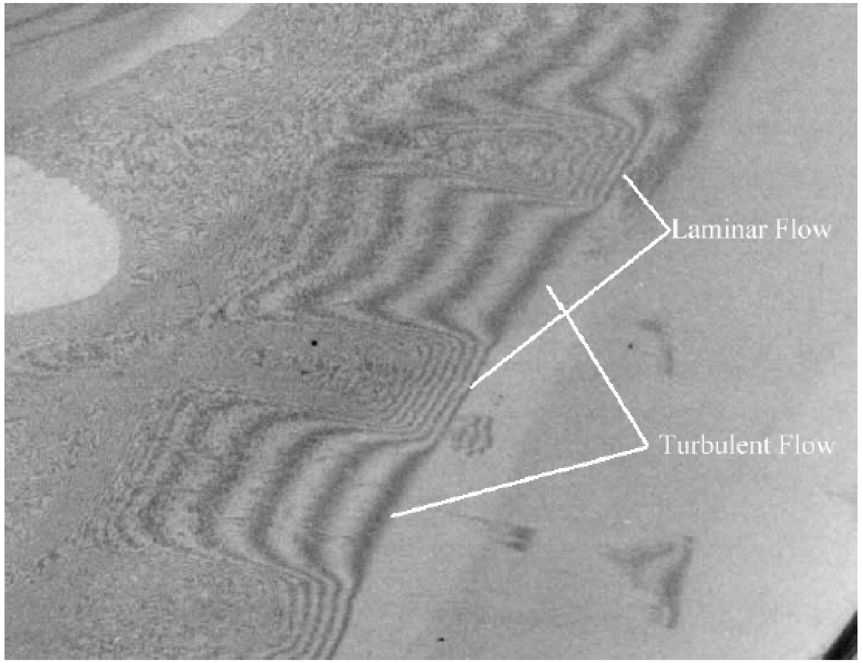


Taken from: Wireless Interconnectivity and Control of Active Systems (WICAS) Website

Taken from: "Microelectromechanical systems-based feedback control of turbulence for skin friction reduction," Kasagi et al. 2009

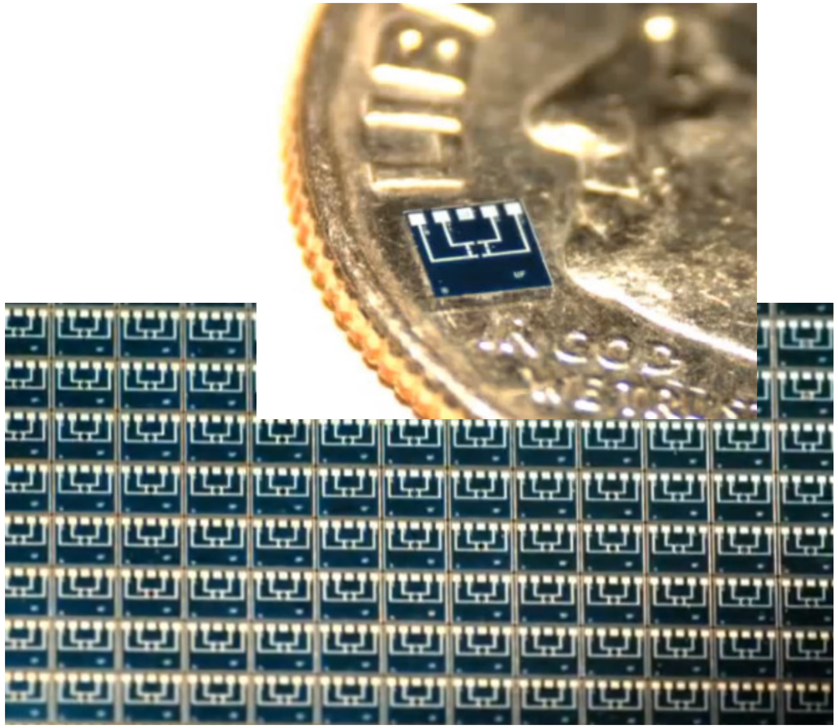
Active Flow Control

Picture of a pressure distribution over an aircraft wing (present well established patterns)



Interferograms of an oil film thinning beneath a turbulent boundary layer. Naughton et al. 2002

Sensing technology is already available

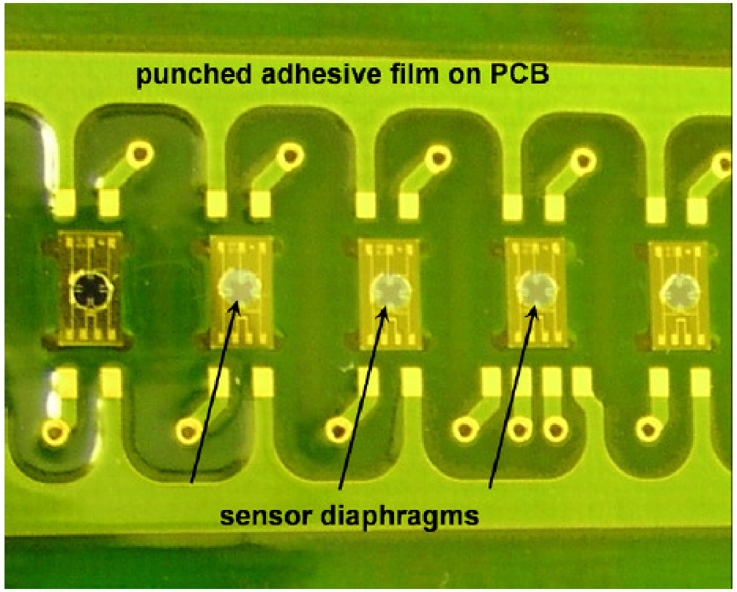


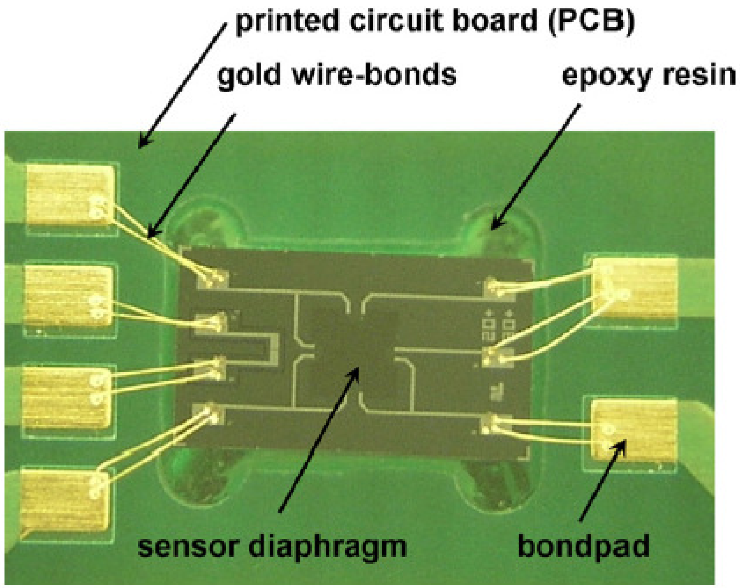
Hot wire shear stress sensor from FCAAP

Active Flow Control

Sensing technology is already available

High resolution 2.5×4.5×0.3 mm³ pressure sensor chip at 160kHz





A. Berns, U. Buder, E. Obermeier, A. Wolter, and A. Leder, "AeroMEMS sensor array for high-resolution wall pressure measurements," *Sensors and Actuators A: Physical*, vol. 132, no. 1, pp. 104-111, Nov. 2006.

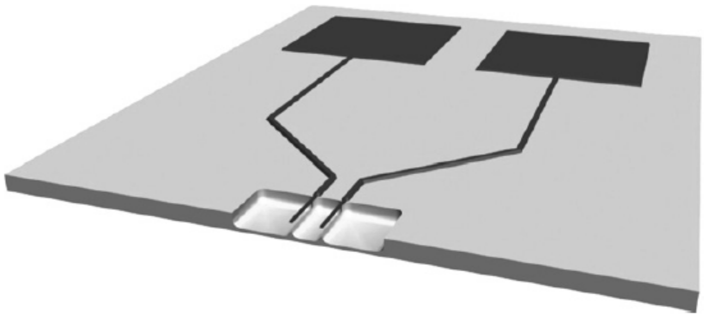
Active Flow Control

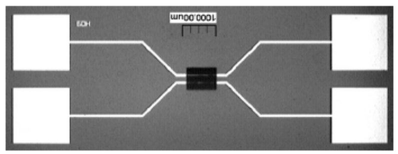
Sensing technology is already available

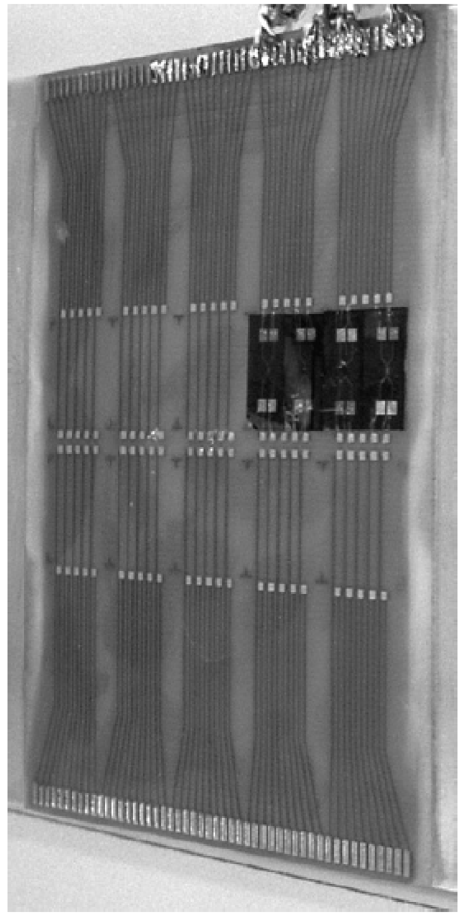
Double Hot-Wire Flow Sensor Dimensions: 800×600µm2

It indirectly measures flow speed, by measuring the heat transfer between two wires.

Signal processing and calculations are required to extract the desirable variable





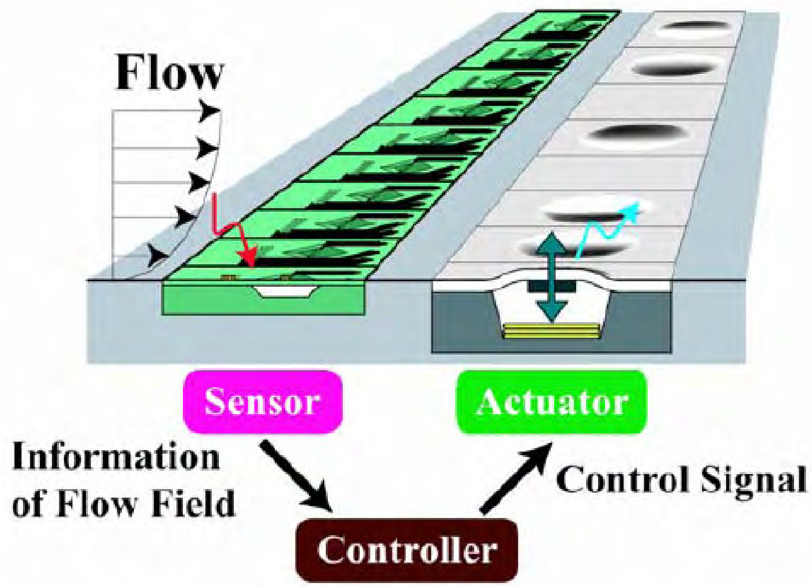


U. Buder, R. Petz, M. Kittel, W. Nitsche, and E. Obermeier, "AeroMEMS polyimide based wall double hot-wire sensors for flow separation detection," *Sensors and Actuators A: Physical*, vol. 142, no. 1, pp. 130-137, Mar. 2008.

Active Flow Control

Active flow control: consists of manipulating a flow to affect a desired change in a closed loop

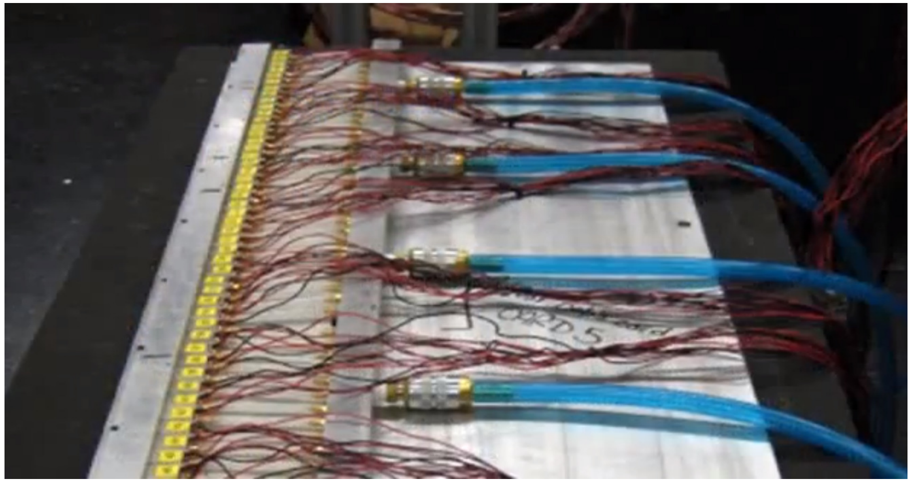
Depending on the technology, one or more sensors provide data for the actuation of one or more actuators.

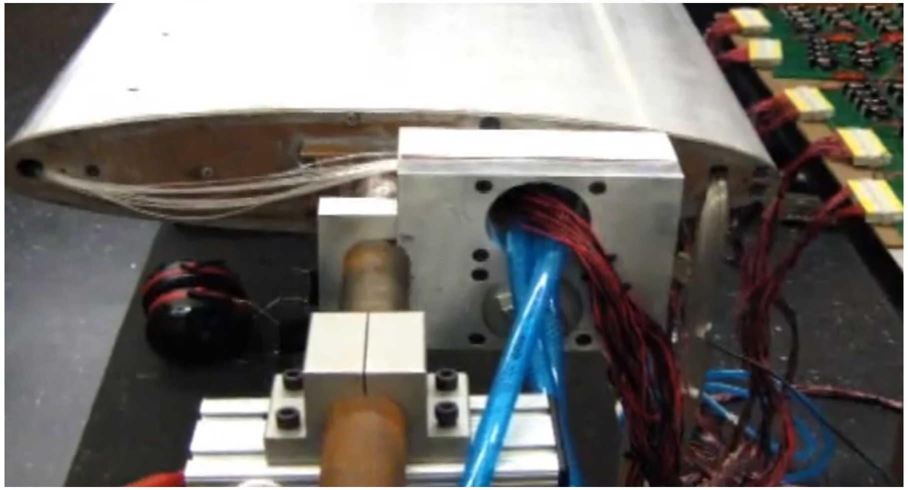


Active Flow Control

The controller is usually centralized, external to the deployment

Wiring can be prohibitive





Biomedical Devices

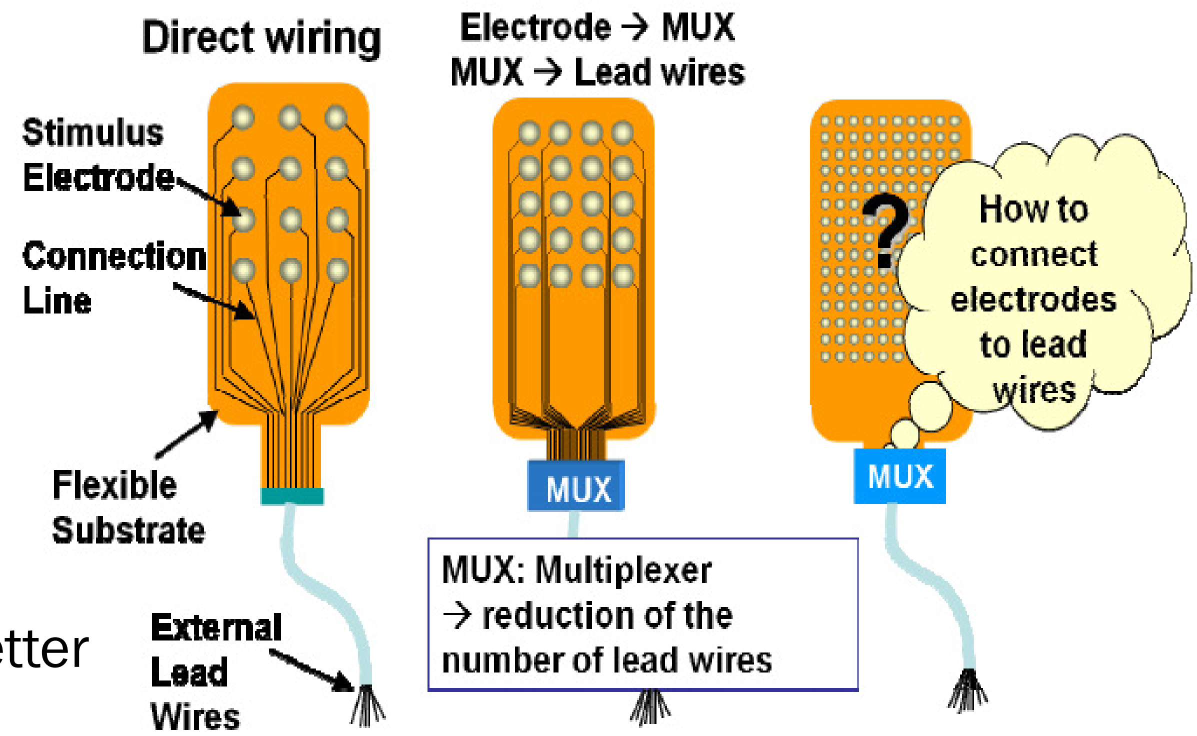
What Applicatio

Biomedical Devices

Retinal prosthesis and brain implantable devices

Electrodes are utilized to stimulate the brain, providing visual perception to blind people

Higher the granularity, better



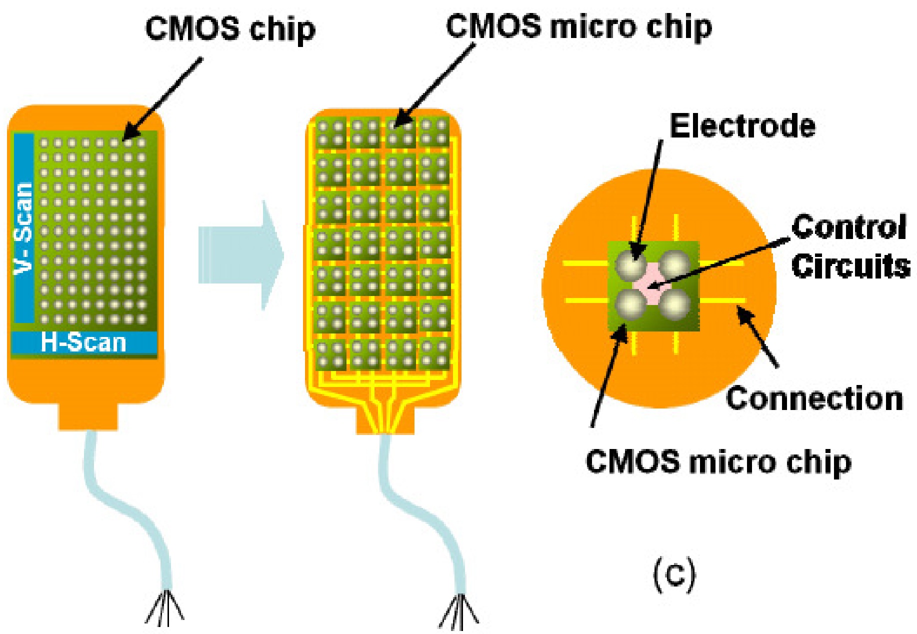
Ohta, Jun, et al. "Implantable CMOS biomedical devices." Sensors (Basel, Switzerland) 9.11 (2009): 9073.

Biomedical Devices

Retinal prosthesis and brain implantable devices

A network is required

Shared buses (I2C like) and centralized V-H scanners are utilized

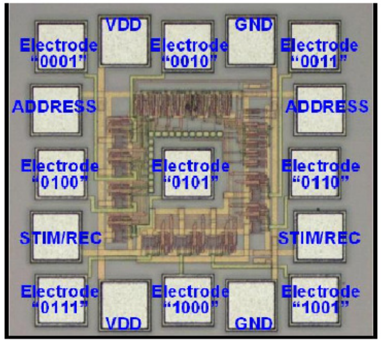


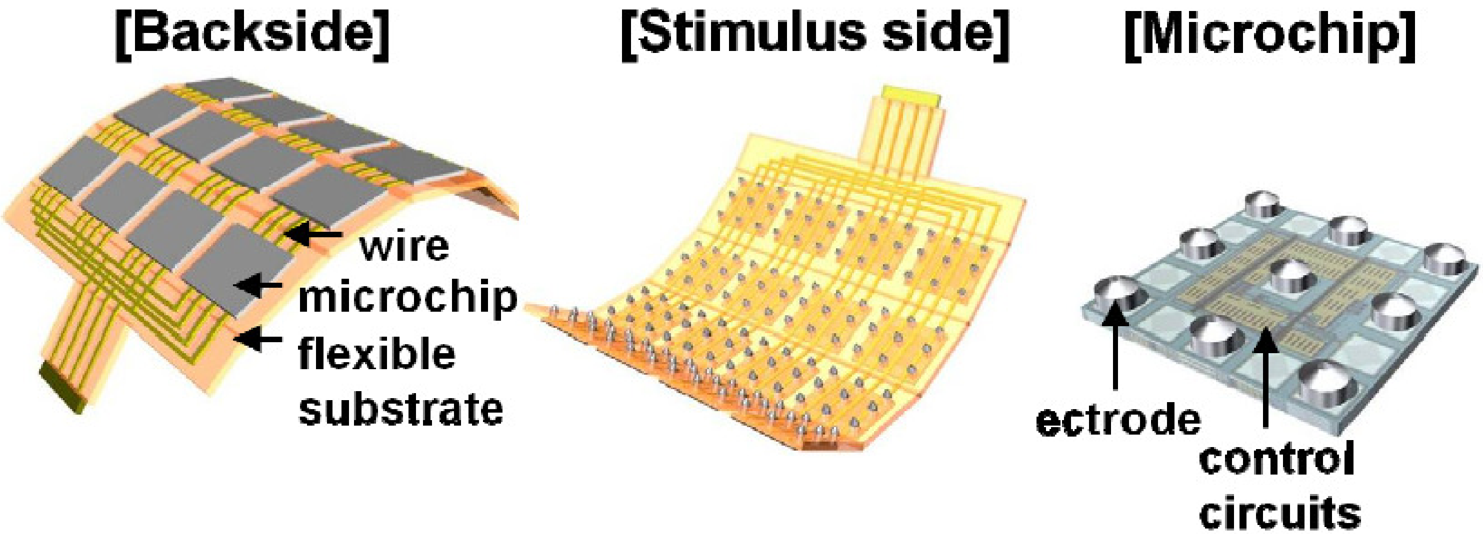
Ohta, Jun, et al. "Implantable CMOS biomedical devices." Sensors (Basel, Switzerland) 9.11 (2009): 9073.

Biomedical Devices

Retinal prosthesis and brain implantable devices

Each set of electrodes is have one digital controller, individually addressable in the network





Ohta, Jun, et al. "Implantable CMOS biomedical devices." Sensors (Basel, Switzerland) 9.11 (2009): 9073.

Robotics

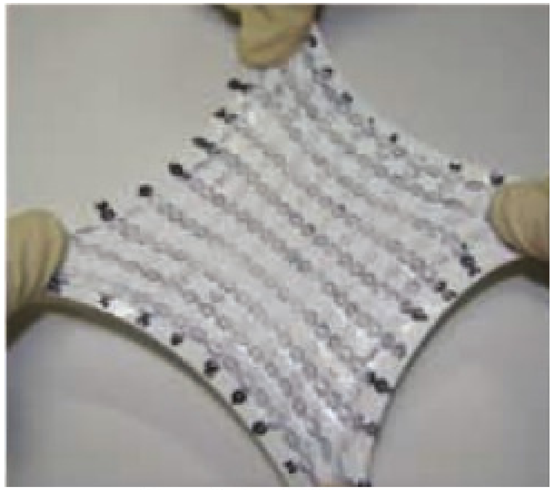
What Applicatio

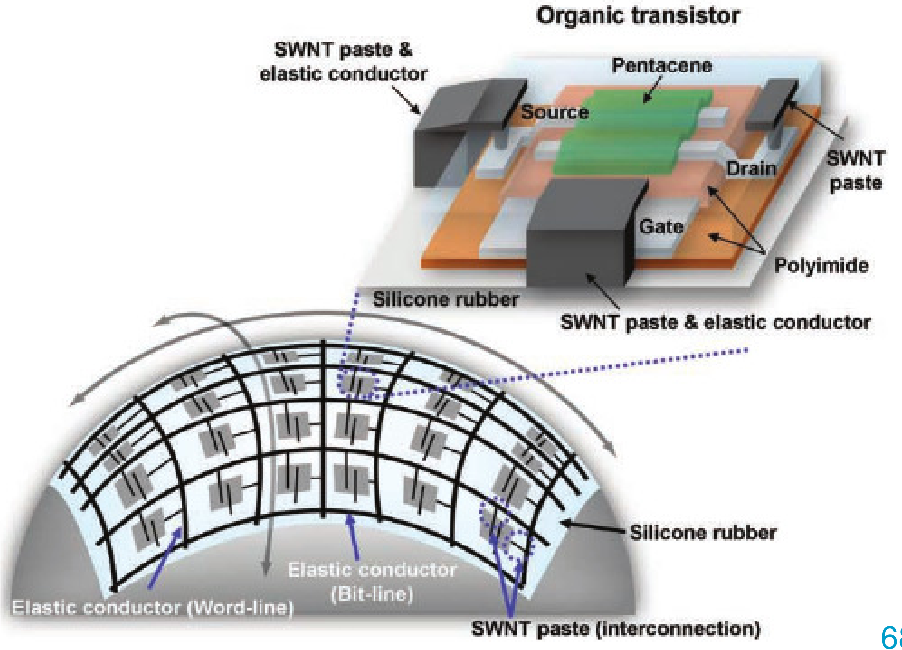
Robotics

- e-Skins

Can enable sensing of quantities of all kinds: magnetic, capacitive, temperature, light, etc..

An efficient way of extracting data is required





Sekitani, Tsuyoshi, et al. "A rubberlike stretchable active matrix using elastic conductors." Science 321.5895 (2008): 1468-1472.

More Sophisticated Processing

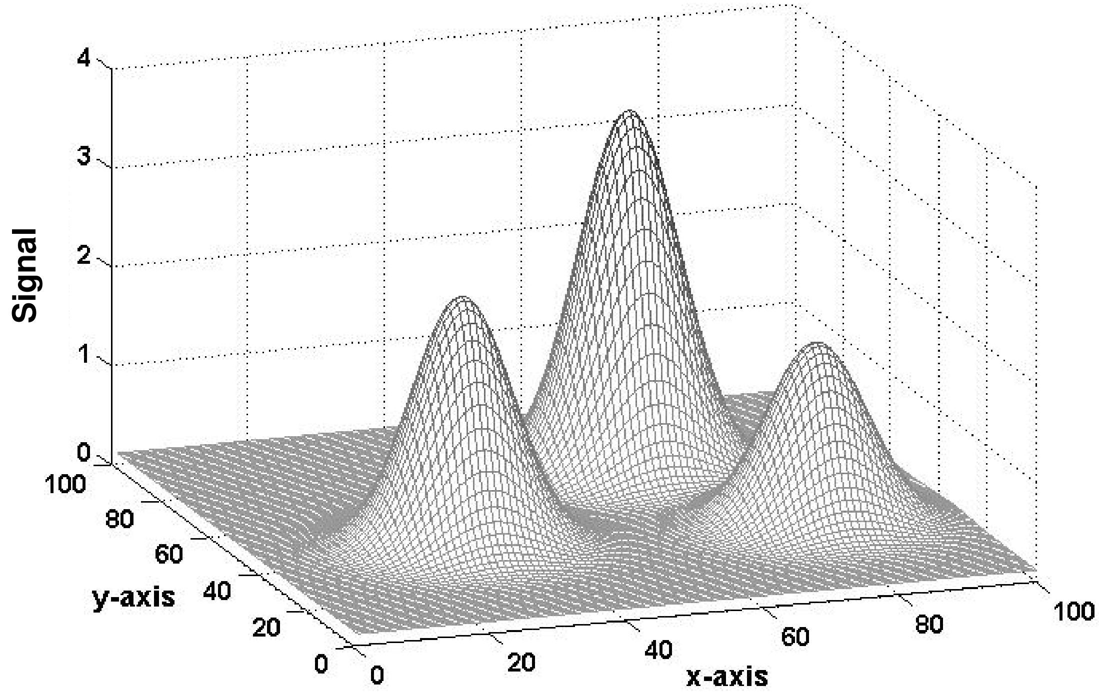
What Applicatio

Feature Extraction

- feature extraction (1)
  - consider a signal (say concentration of a hazardous gas) that varies with location (x,y)

how can we define number of peaks and the boundary around each one?

(assume a deployment of a dense network of sensing nodes)

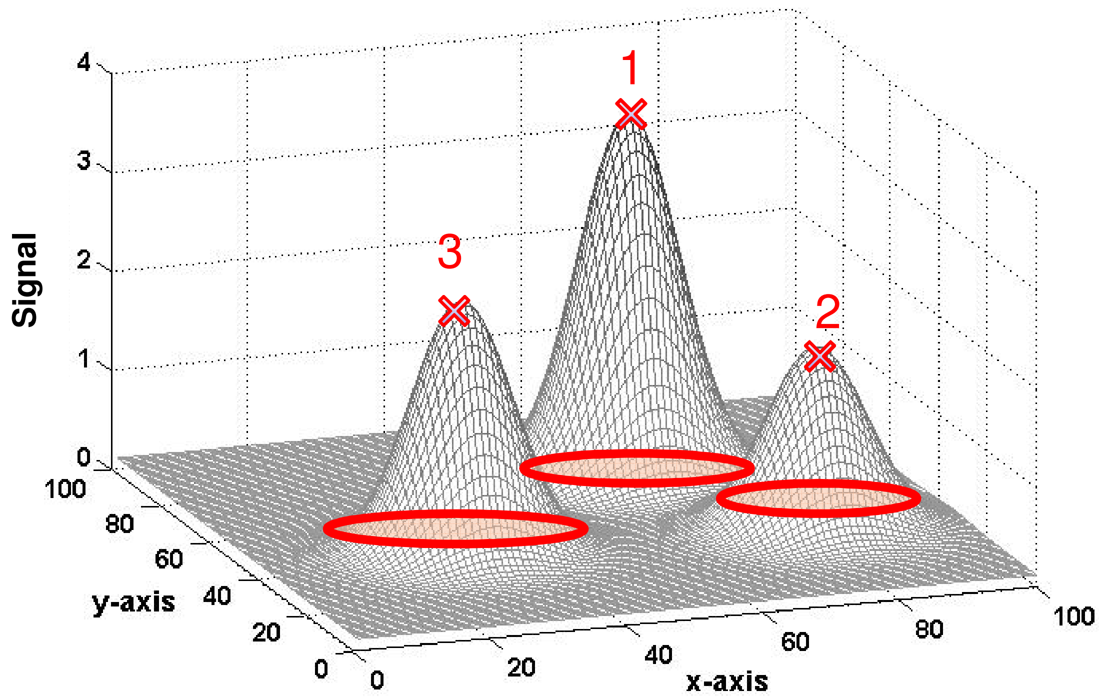


Feature Extraction

- feature extraction (2)
  - consider a signal (say concentration of a hazardous gas) that varies with location (*x*,*y*)

how can we define number of peaks and the boundary around each one?

one can get sensor readings from **all** the nodes and then perform central data processing…
but it would be slow: O(*m*)

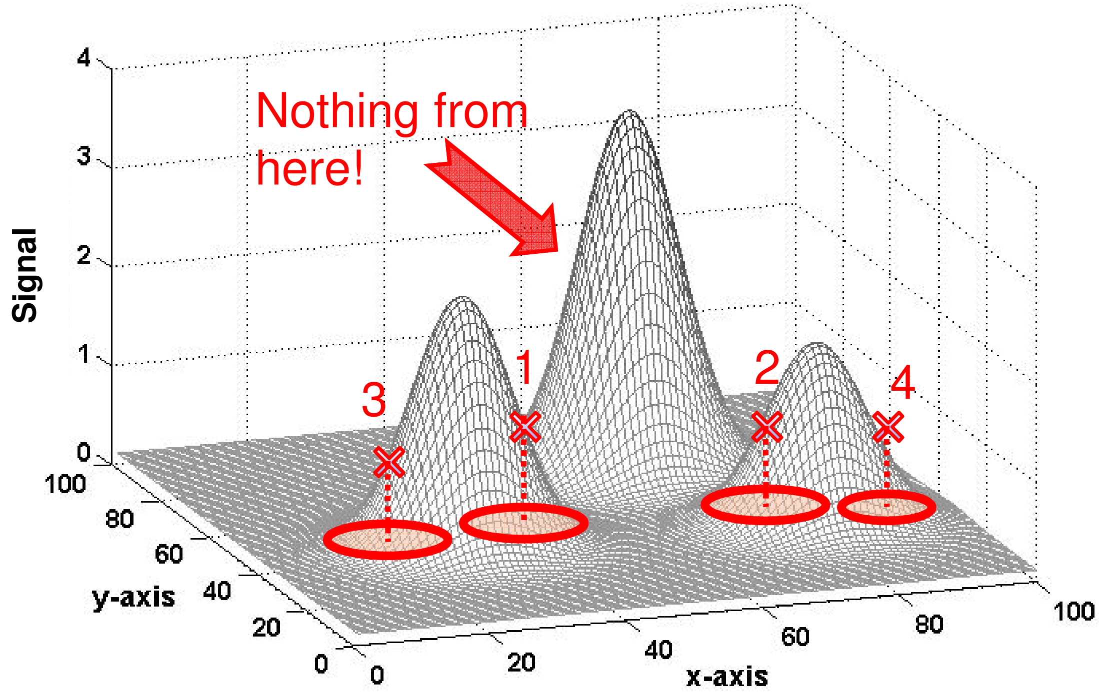


Feature Extraction

- feature extraction (3)
  - consider a signal (say concentration of a hazardous gas) that varies with location (*x*,*y*)

how can we define number of peaks and the boundary around each one?

one can get sensor readings from **some of** the nodes **randomly**…
but it is going to be inaccurate

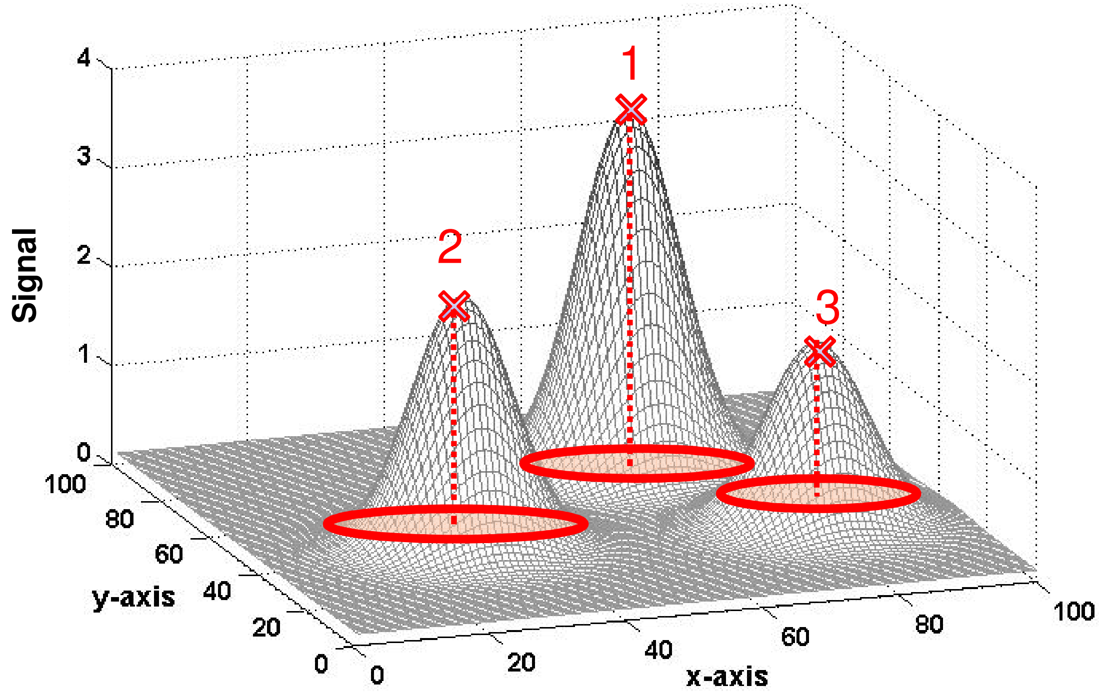


Feature Extraction

- feature extraction (4)
  - consider a signal (say concentration of a hazardous gas) that varies with location (*x*,*y*)

how can we define number of peaks and the boundary around each one?

one can get sensor readings from **some of** the nodes with most **constructive** information … Nice job!



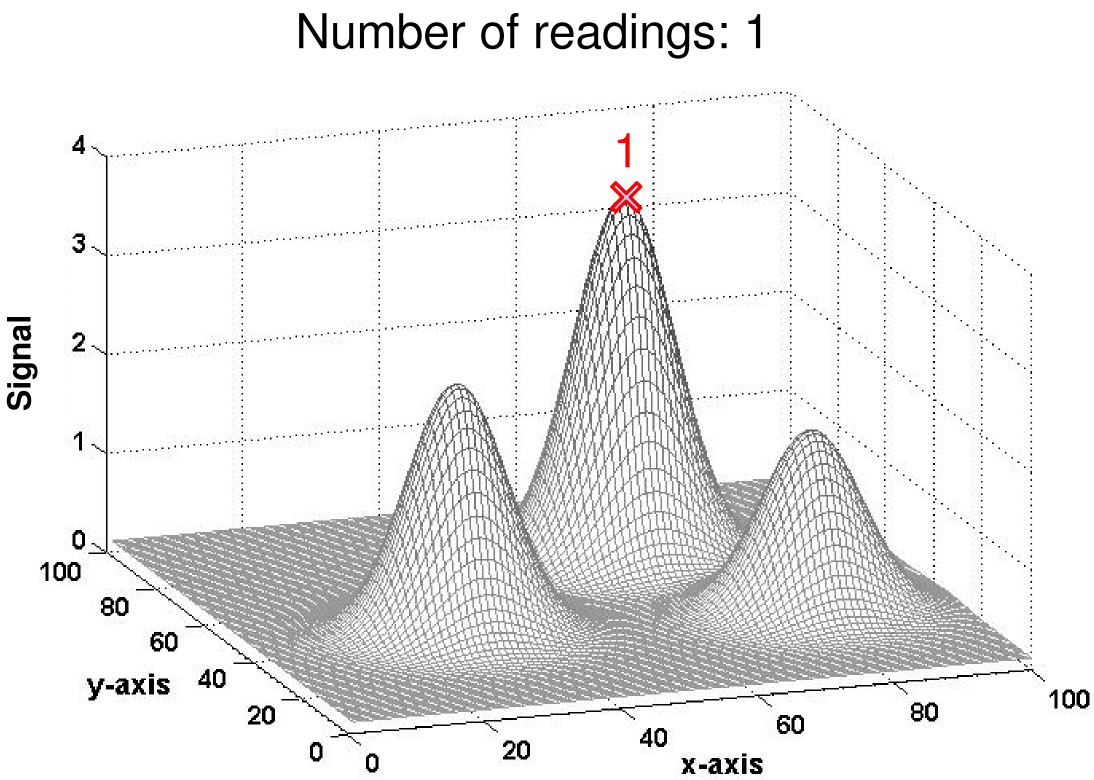
Feature Extraction

- feature extraction (5)
  - consider a signal that varies with location (*x*,*y*)

how to select nodes with most constructive information ?

iteratively select the node with highest reading and then define the node with smallest reading to it.

start with the global MAX



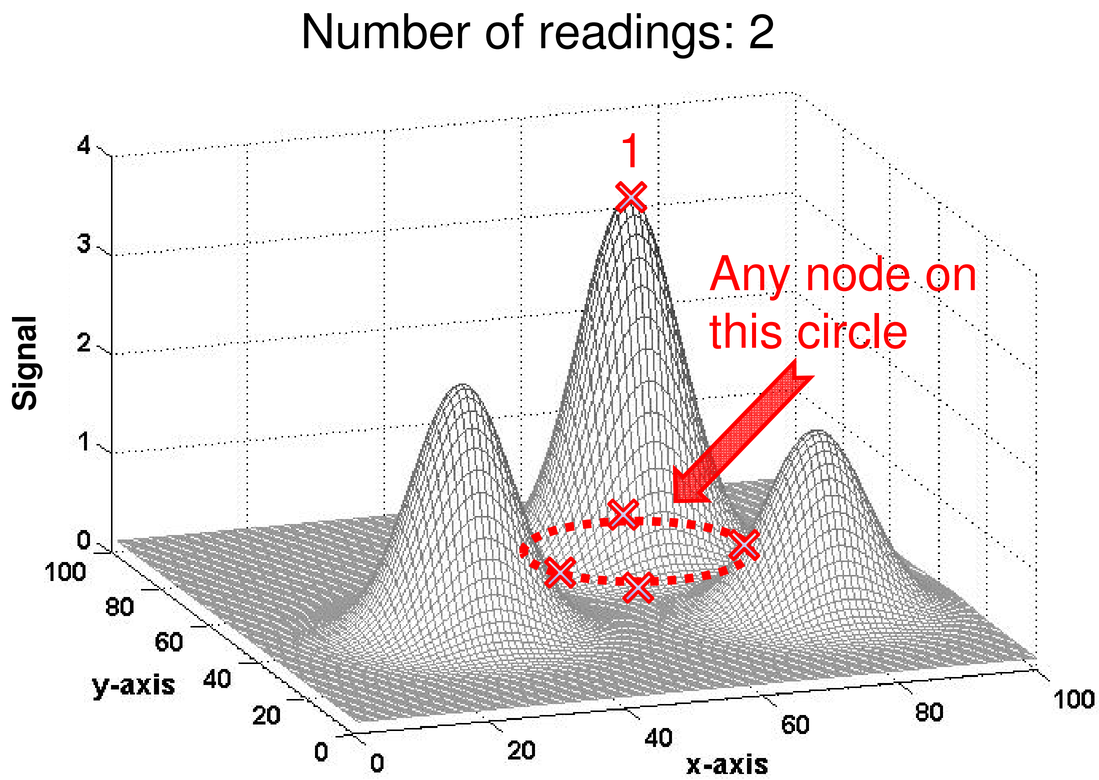
Feature Extraction

- feature extraction (6)
  - consider a signal that varies with location (*x*,*y*)

how to select nodes with most constructive information ?

iteratively select the node with highest reading and then define the node with smallest reading to it.

find the nearest local Min to the defined global MAX



Feature Extraction

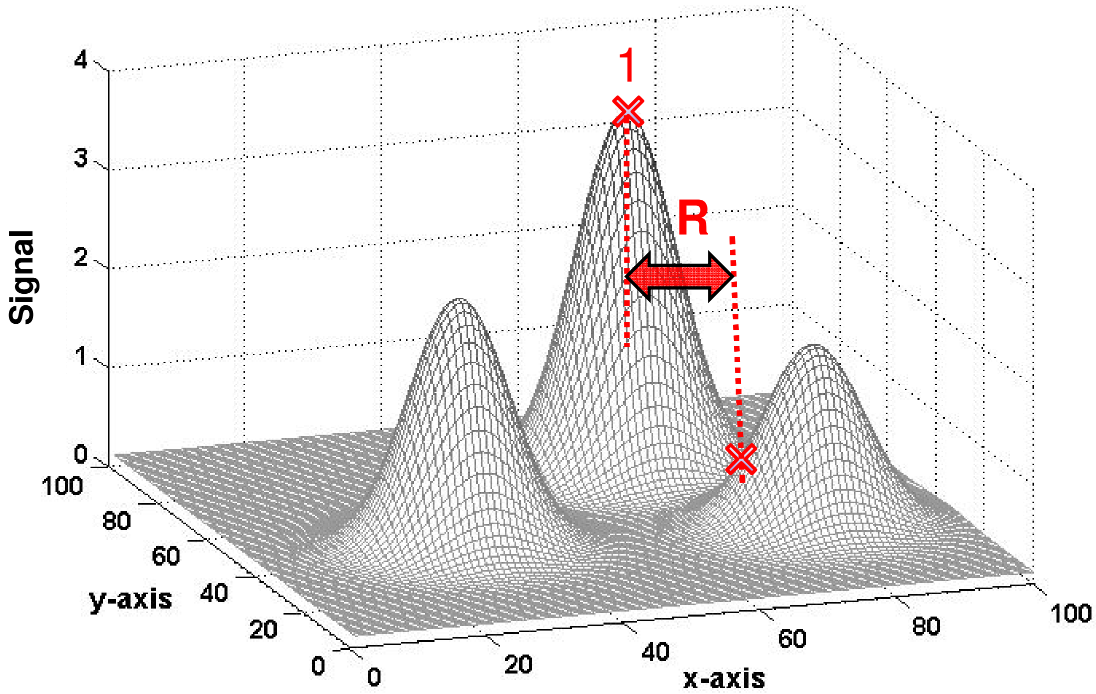
- feature extraction (7)
  - consider a signal that varies with location (*x*,*y*)

how to select nodes with most constructive information ?

iteratively select the node with highest reading and then define the node with smallest reading to it.

determine the radius of the boundary

Number of readings: 2



Feature Extraction

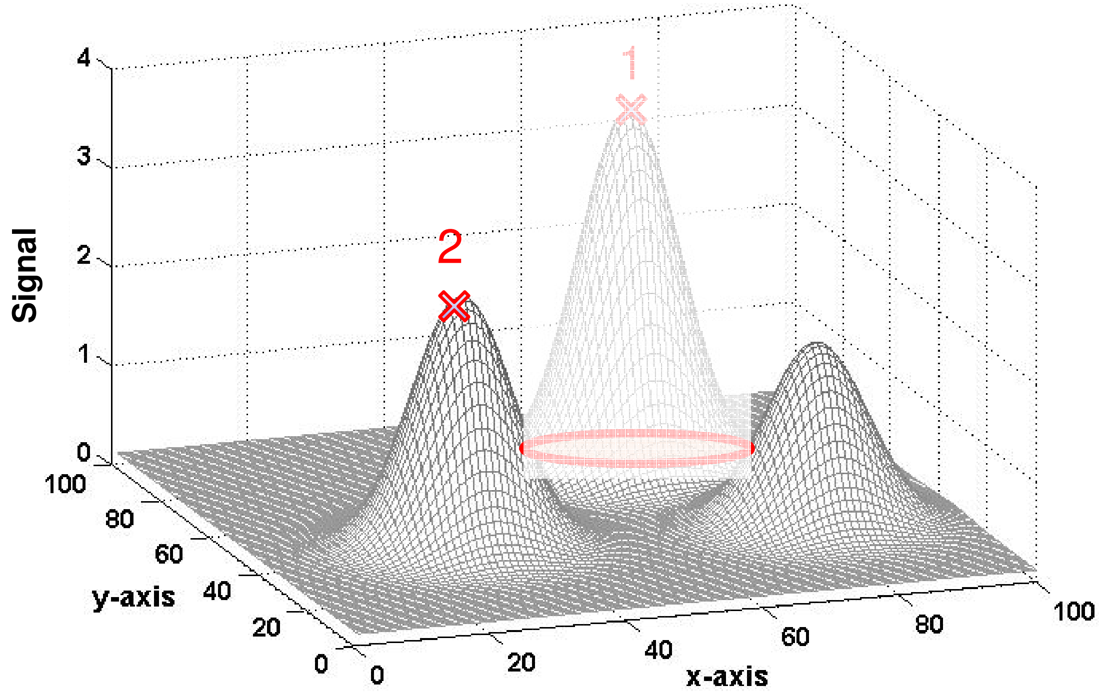
- feature extraction (8)
  - consider a signal that varies with location (x,y)

how to select nodes with most constructive information ?

iteratively select the node with highest reading and then define the node with smallest reading to it.

eliminate nodes located inside the defined boundary and repeat the procedure

Number of readings: 3



Feature Extraction

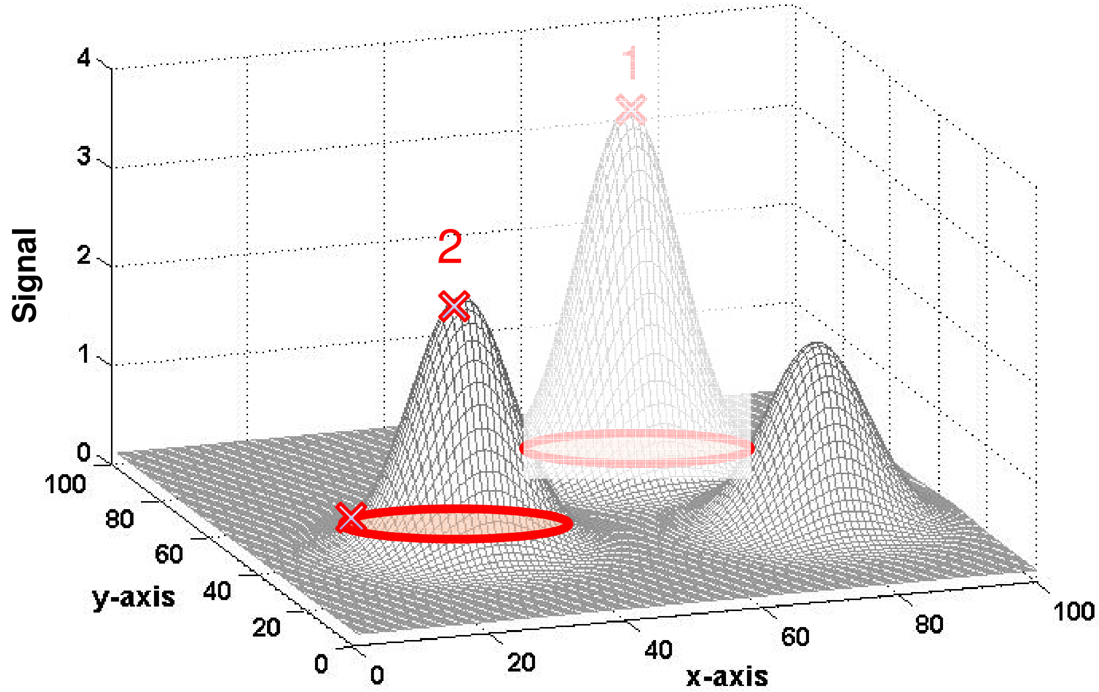
- feature extraction (9)
  - consider a signal that varies with location (x,y)

Number of readings: 4

how to select nodes with most constructive information ?

iteratively select the node with highest reading and then define the node with smallest reading to it.

eliminate nodes located inside the defined boundary and repeat the procedure



Feature Extraction

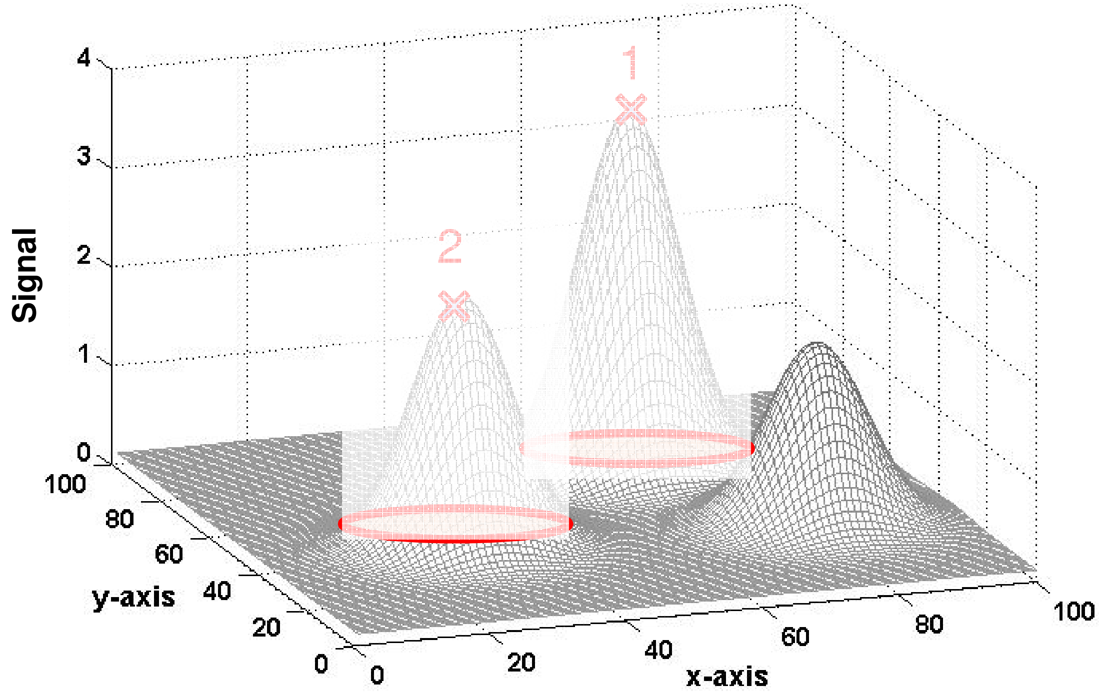
- feature extraction (10)
  - consider a signal that varies with location (*x*,*y*)

how to select nodes with most constructive information ?

iteratively select the node with highest reading and then define the node with smallest reading to it.

eliminate nodes located inside the defined boundary and repeat the procedure

Number of readings: 4



Feature Extraction

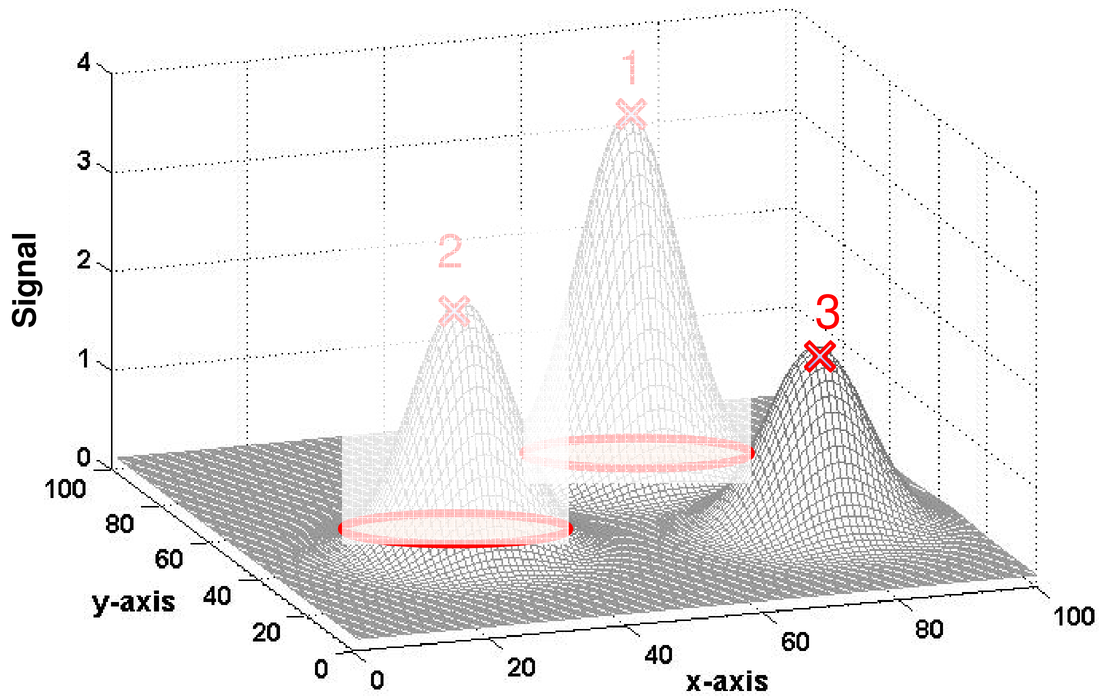
- feature extraction (11)
  - consider a signal that varies with location (x,y)

how to select nodes with most constructive information ?

iteratively select the node with highest reading and then define the node with smallest reading to it.

eliminate nodes located inside the defined boundary and repeat the procedure

Number of readings: 5



Feature Extraction

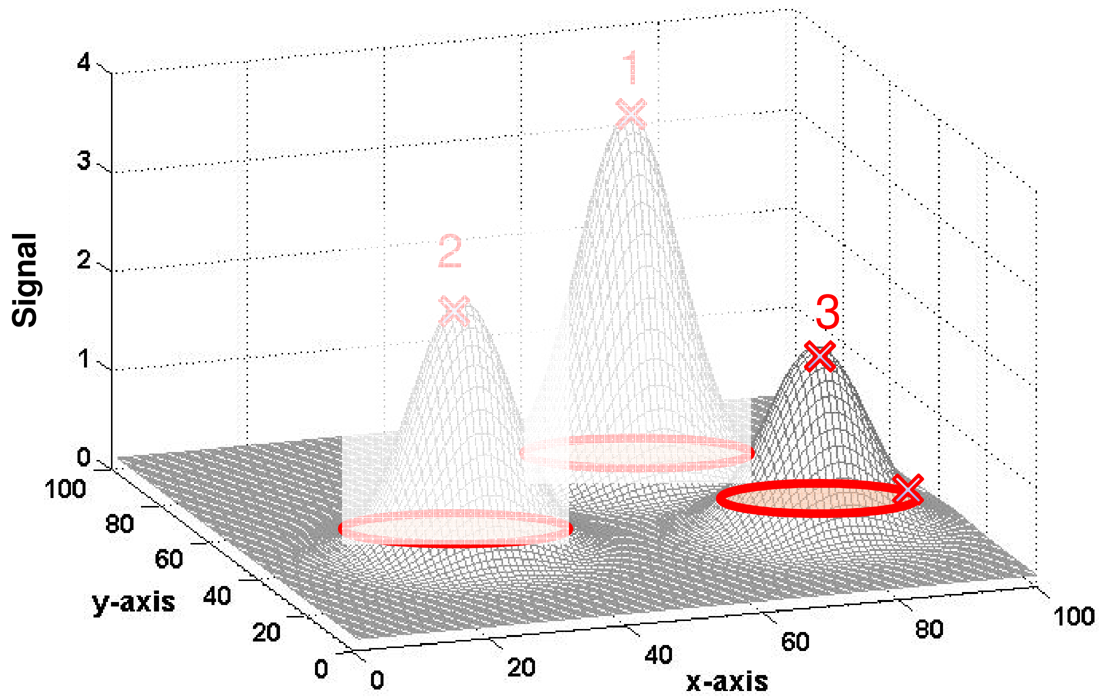
- feature extraction (12)
  - consider a signal that varies with location (*x*,*y*)

Number of readings: 6

how to select nodes with most constructive information ?

iteratively select the node with highest reading and then define the node with smallest reading to it.

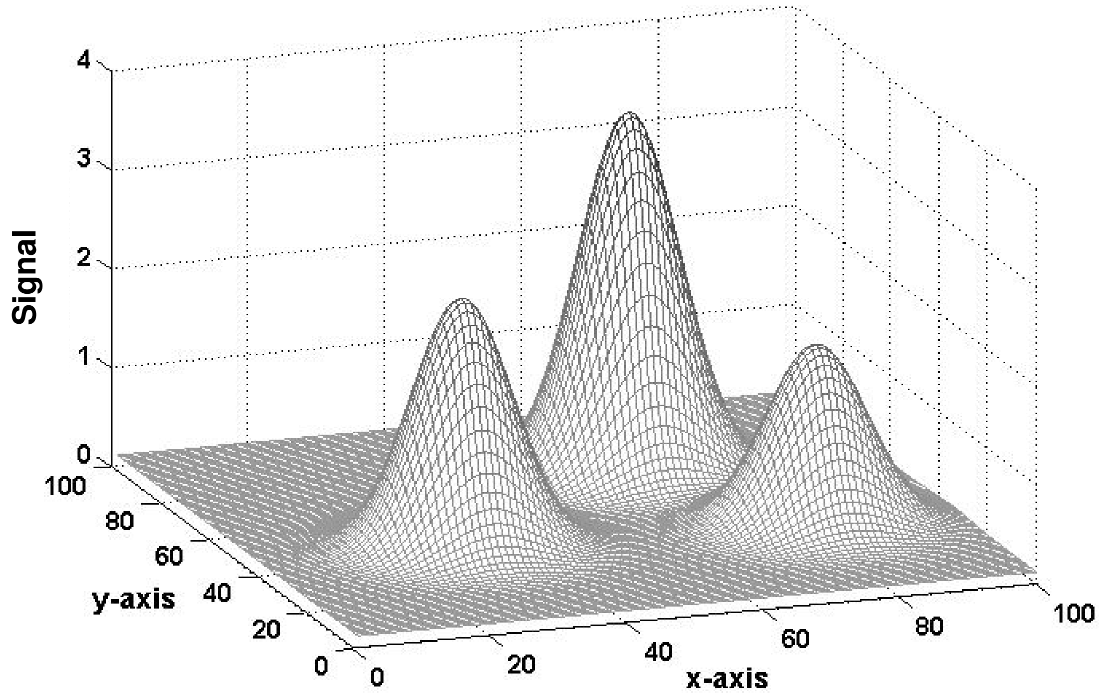
eliminate nodes located inside the defined boundary and repeat the procedure



Feature Extraction

- feature extraction (13)
  - consider a signal that varies with location (*x*,*y*)

the (PD)² protocol can be exploited to efficiently select the nearest local minimum!



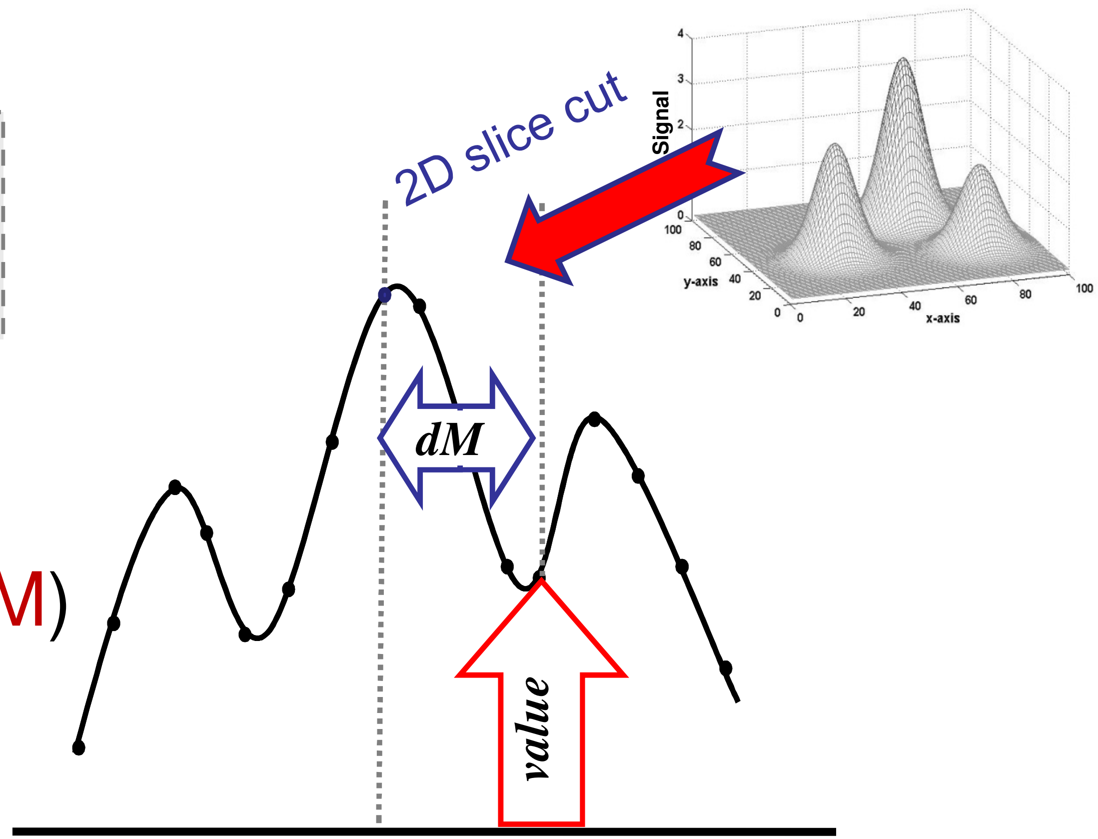
Feature Extraction

- feature extraction (14)
  - consider a signal that varies with location (x,y)

the (PD)^2 protocol can be exploited to efficiently select the nearest local minimum!

Joint minimization

Priority: f(value x dM)



Feature Extraction

- feature extraction (15)
  - consider a signal that varies with location (x,y)

Is it possible to find the boundary around all peaks?

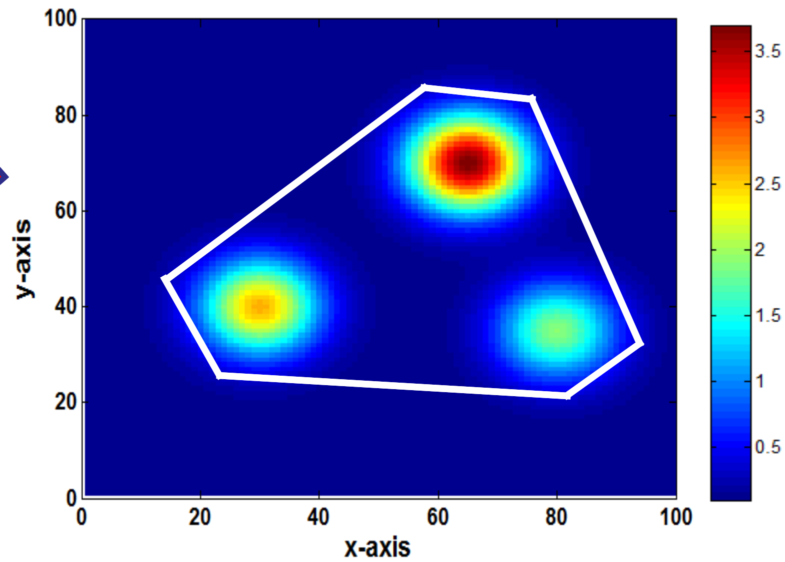
what if signal is not distributed evenly around the peak?

Feature Extraction

- feature extraction (16)
  - consider a signal that varies with location (*x*,*y*)

Is it possible to find the boundary around all peaks?

what if signal is not distributed evenly around the peak?



Priority: $f(\text{value} \times \text{location} \times \vec{u})$

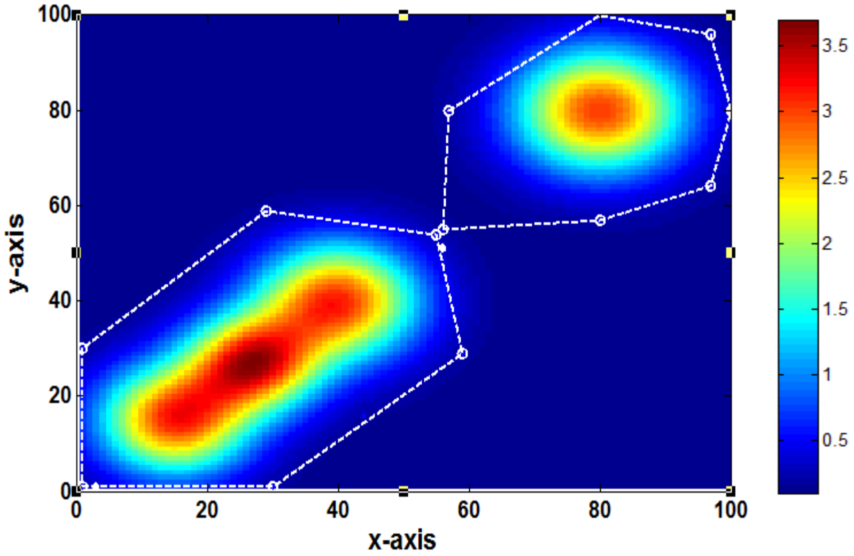
Where $\vec{u}$ is a vector

Feature Extraction

- feature extraction (17)
  - consider a signal that varies with location (*x*,*y*)

Is it possible to find the boundary around all peaks?

what if signal is not distributed evenly around the peak?



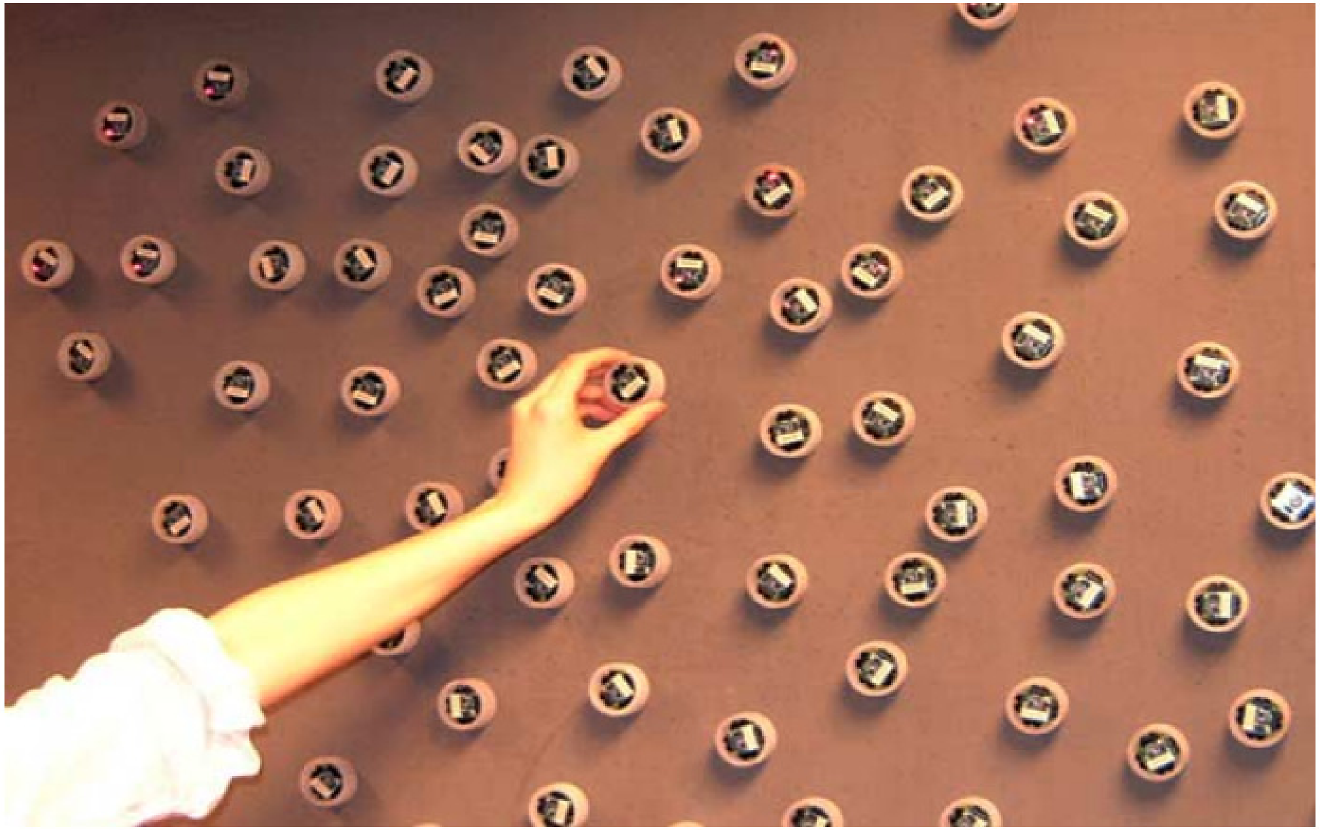
Priority: $\boldsymbol{f}$(value x location x dM x $\vec{u}$)   Where $\vec{u}$ is a vector

DCOSS 2014 Paper

That's All Folks!

Implementation

- **Pushpin** Sensors
  - Plug-n-Play



Lifton, Joshua, et al. "Pushpin computing system overview: A platform for distributed, embedded, ubiquitous sensor networks." Pervasive Computing. Springer Berlin Heidelberg, 2002. 139-151.