Virtual Sensors: Abstracting Data from Physical Sensors

Sanem Kabadayı, Adam Pridgen, and Christine Julien
Mobile and Pervasive Computing Group
The University of Texas at Austin

June 26, 2006

Overview

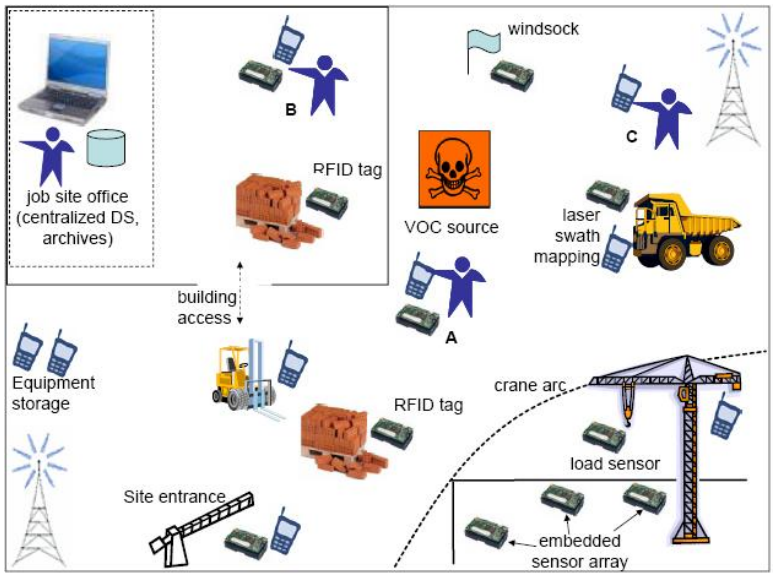
- Motivation and Challenges
- What are Virtual Sensors?
- Virtual Sensor Model
- Middleware Supporting Virtual Sensors
- Evaluating an Example Virtual Sensor
- Conclusion

Motivation

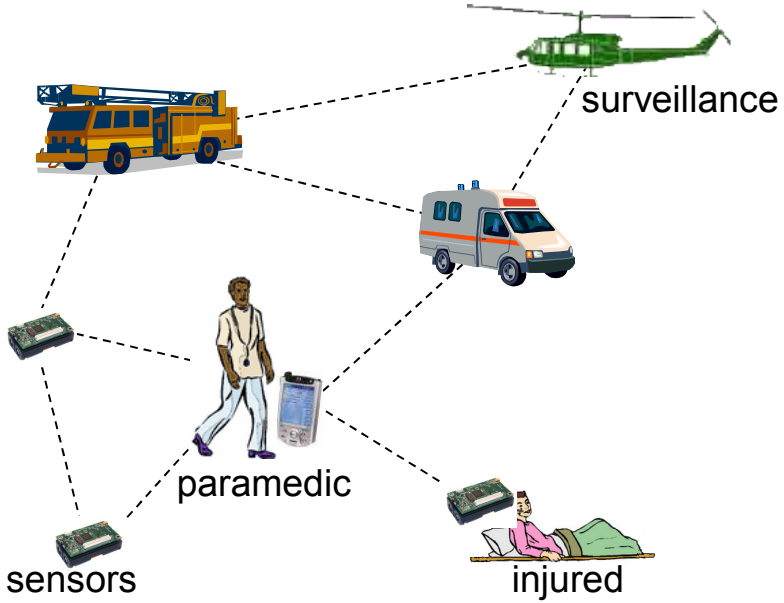
- Sensor networks
  - Are rapidly becoming ubiquitous in pervasive computing environments
  - Consist of miniature, battery-powered devices that monitor the instrumented environment
- Many potential applications
  - Aware homes, intelligent construction sites, first-responder deployments, battlefield scenarios
- In emerging scenarios, need to extract abstracted measurements from diverse sensors

Application Examples

Intelligent construction site

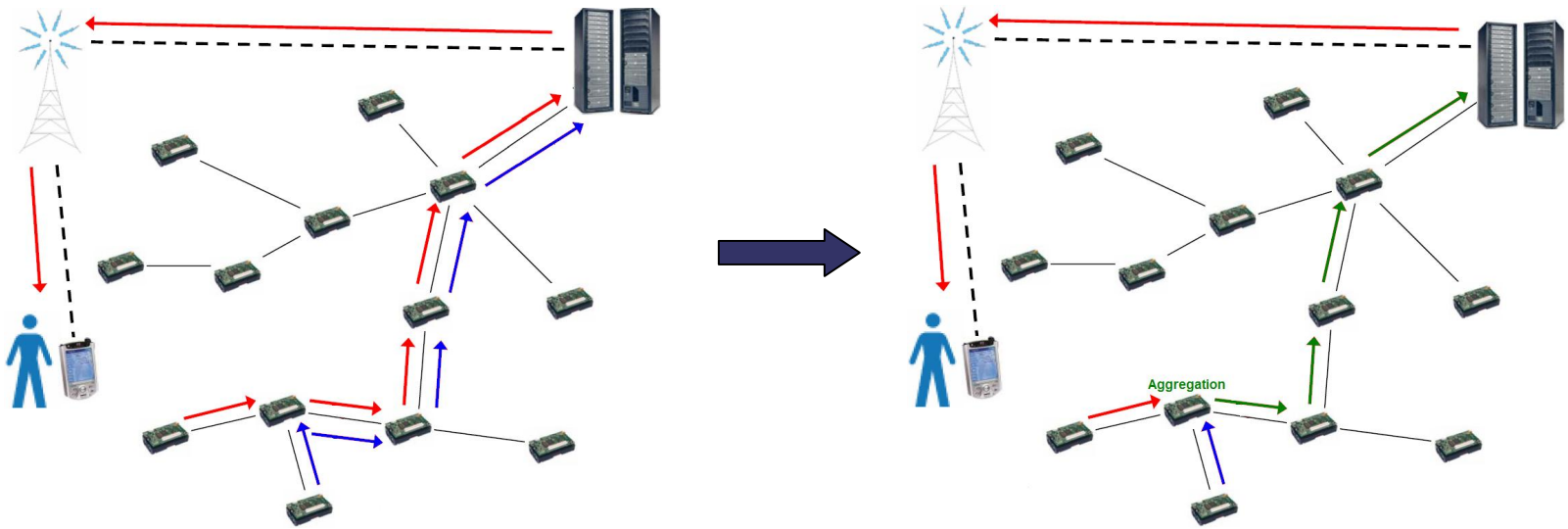


First-responder deployment



Challenges

- Existing deployments require sensors to relay raw information to sink nodes for further processing
  - Inefficient battery and bandwidth usage



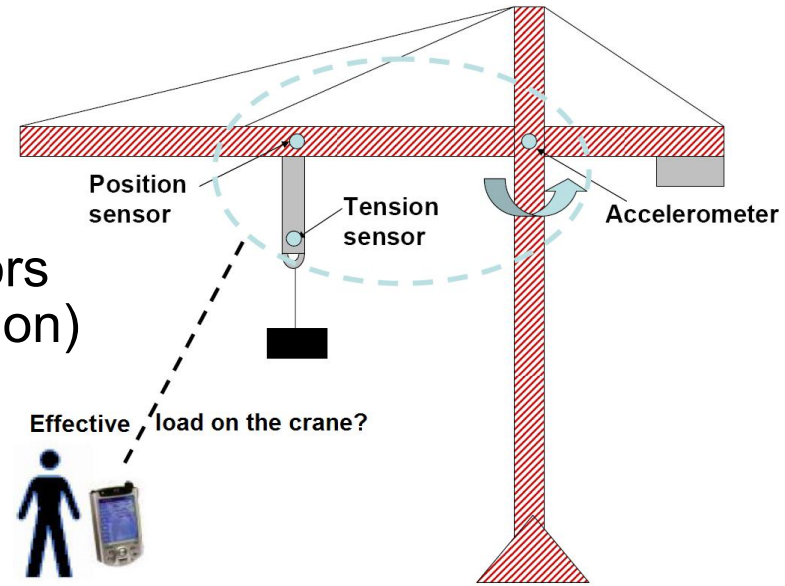
- In-network data processing algorithms support only standard mathematical operators (MIN, COUNT, AVG) over *homogeneous* data
- Reusability

Virtual Sensors

- Software sensor, NOT a physical or hardware sensor
- Provide indirect measurements of abstract conditions
  - By combining sensed data from group of *heterogeneous* physical sensors
- Allow application developer to create sensors that do not exist in the physical world

Virtual Sensor Example

- Safe load indicator on a crane
  - Uses physical sensors that monitor boom angle, load, telescoping length, wind speed, etc.
- Software implementing this virtual sensor may
  - run on user's handheld device
  - be deployed to one of the sensors (to reduce network communication)



Benefits of Virtual Sensors

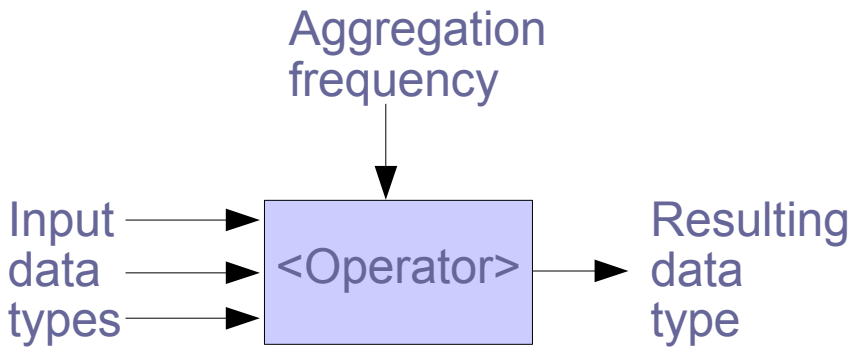
- Reusability
  - Future will see multipurpose networks deployed to support numerous applications
  - Ability to remotely program sensor networks for particular applications essential
- Heterogeneity
  - Power of virtual sensors lies in the ability to use heterogeneous physical sensors
- Masking *explicit* data sources (sensors)
  - A virtual position sensor that
    - uses GPS when available on the local device, *but*
    - can switch to providing a position estimate based on positions of other nearby (physical) location sensors if GPS unavailable.

Virtual Sensor Model (1)

- A novel and powerful programming abstraction
  - Allows the programmer to simply specify heterogeneous aggregation and subsequently delegate that aggregation to the network
  - Puts aggregation task firmly in the expert's hands, but leaves complex communication programming to the middleware
- An application's high-level specification of a virtual sensor (Java) is translated into low-level code (nesC)
- This code can either run locally or be deployed to a sensor

Virtual Sensor Model (2)

- Client application runs with the support of the virtual sensor abstraction
- Application delegates sensor discovery to virtual sensor
- Declarative specification hides changes in data sources from the application
- The programmer specifies the following four parameters to create the virtual sensor:
  - Input data types
  - Operator (or Aggregator)
  - Resulting data type
  - Aggregation frequency

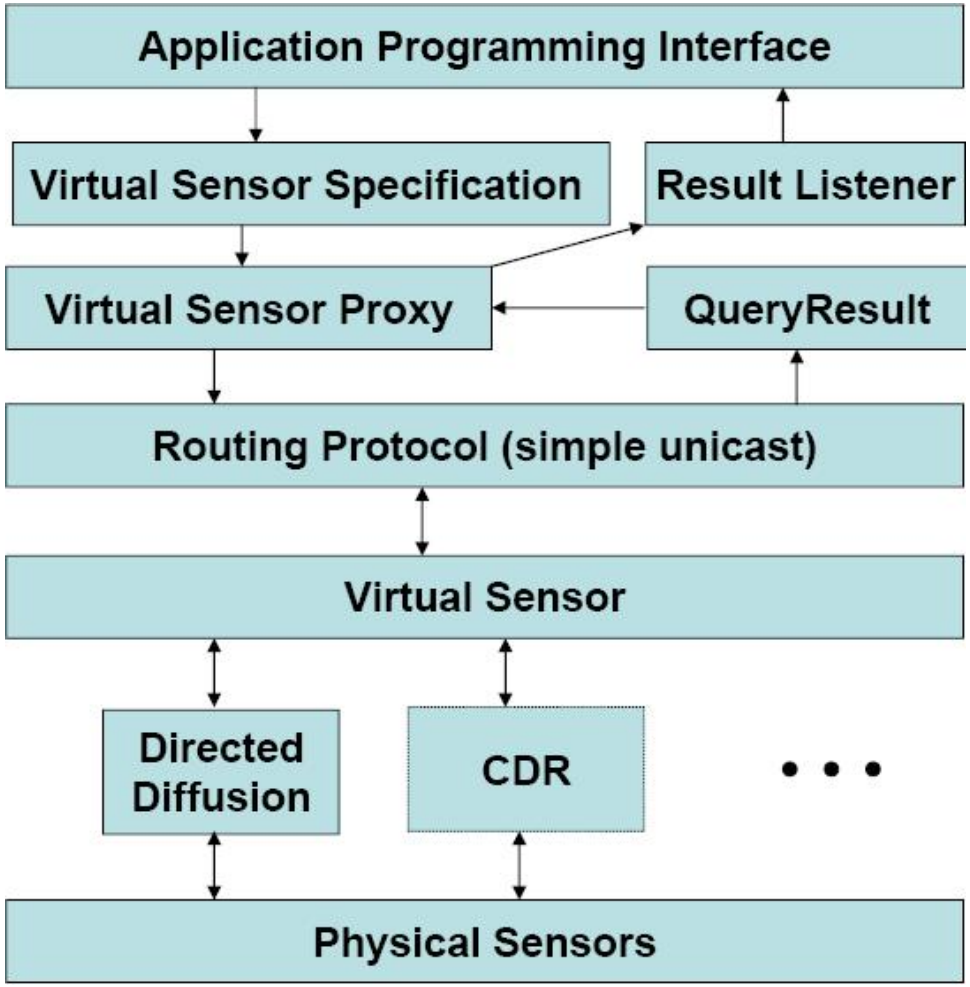


Interacting with Virtual Sensors

- Types of queries enabled on a virtual sensor:
  - One-time
  - Persistent
- Two different methods for posing queries:
  - query()
  - register()

Operation	
`public VirtualSensor(DataType[] inputs, Aggregator a, DataType result, int aggfreq)`	Constructor aggfreq impacts how up-to-date readings are
`void query(ResultListener r)`	Sends one-time query Result listener receives results
`int register(ResultListener r, int reqfreq)`	Registers persistent query Request frequency indicates how often application demands data value from virtual sensor, returns receipt to cancel registration
`void deregister(int receipt)`	Stops the registered query referenced by receipt

Middleware Supporting Virtual Sensors

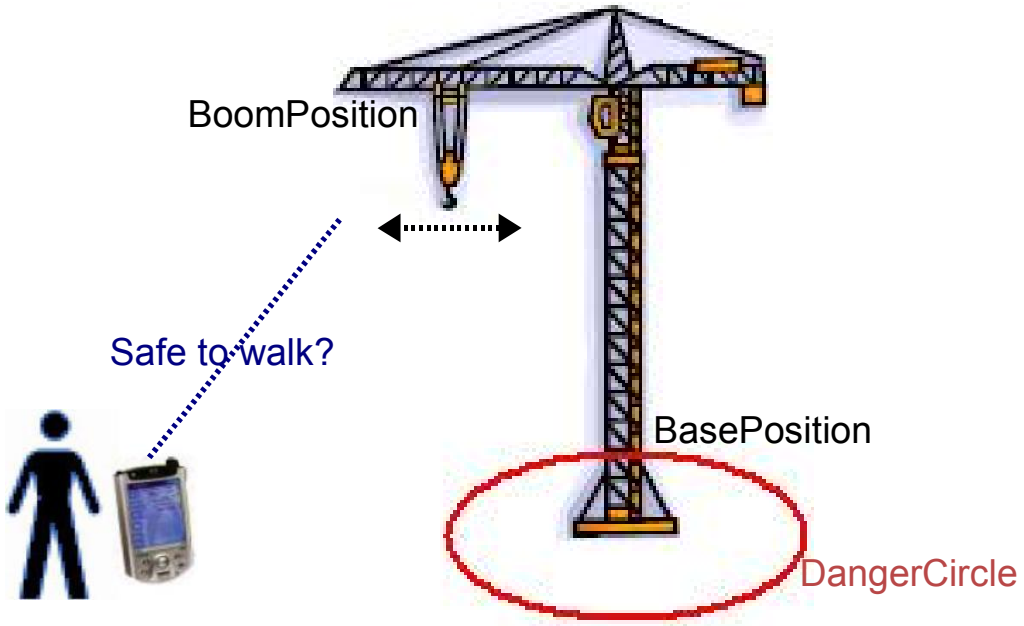


Communication to the software sensor, if it is running on a remote host

Communication among physical sensors that contribute to the virtual sensor

Example Virtual Sensor

- CraneDangerCircle
  - Represents the area near a crane where it is unsafe to walk
  - Centered at the base of the crane (which may move)
  - Has a radius defined by the position of the boom (which is even more likely to move).
  - Expands and contracts accordingly, as the boom moves along the crane arm.



Virtual Sensor Example

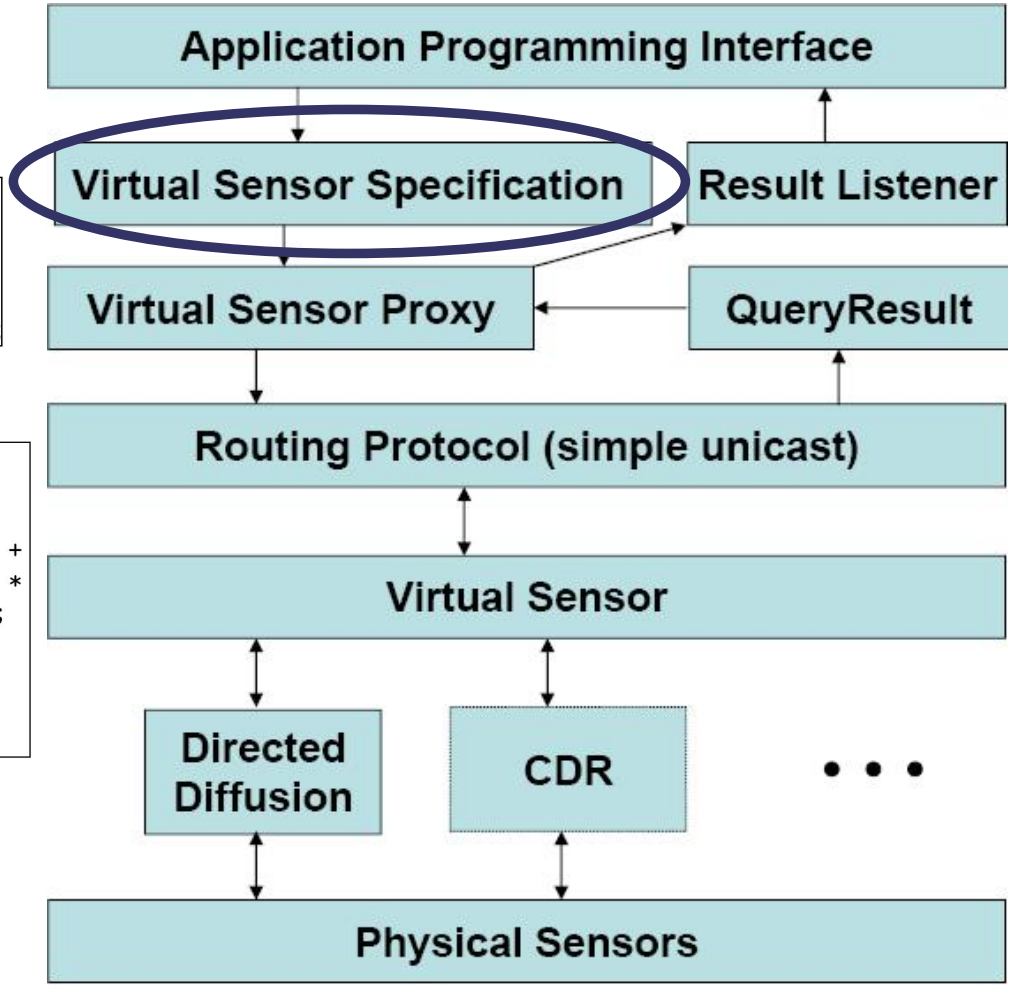
Developer

- Constructs and deploys virtual sensor using knowledge of available data types

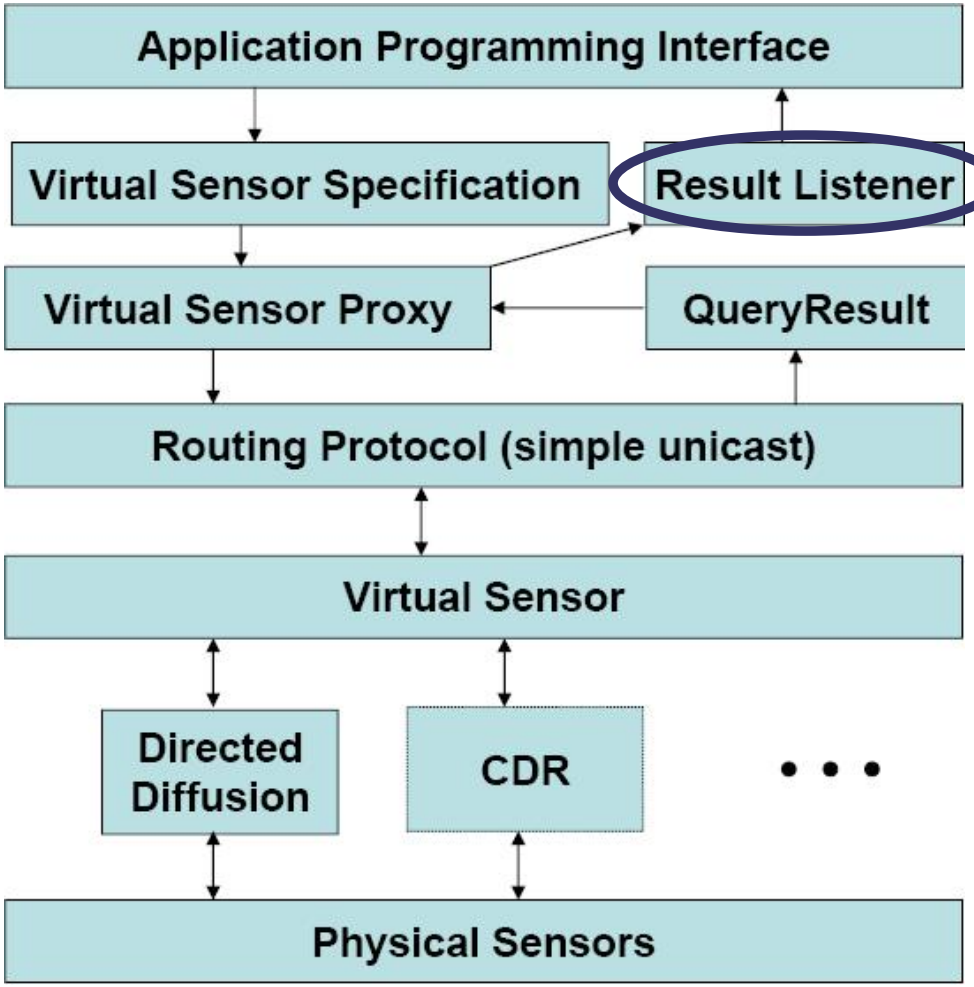
```
VirtualSensor craneVS =
 new VirtualSensor({``BasePosition'',
                    ``BoomPosition''},
                   new CraneAggregator(),
                   ``CraneDangerCircle'');
```

- Specifies mechanics behind aggregator

```
class CraneAggregator implements Aggregator {
  CraneDangerCircle aggregate(DataType[] inputs){
    int radius_squared = (input[0].x - input[1].x) *
                         (input[0].x - input[1].x) +
                         (input[0].y - input[1].y) *
                         (input[0].y - input[1].y);
    return new CraneDangerCircle(input[0],
                                 radius_squared)
  }
}
```



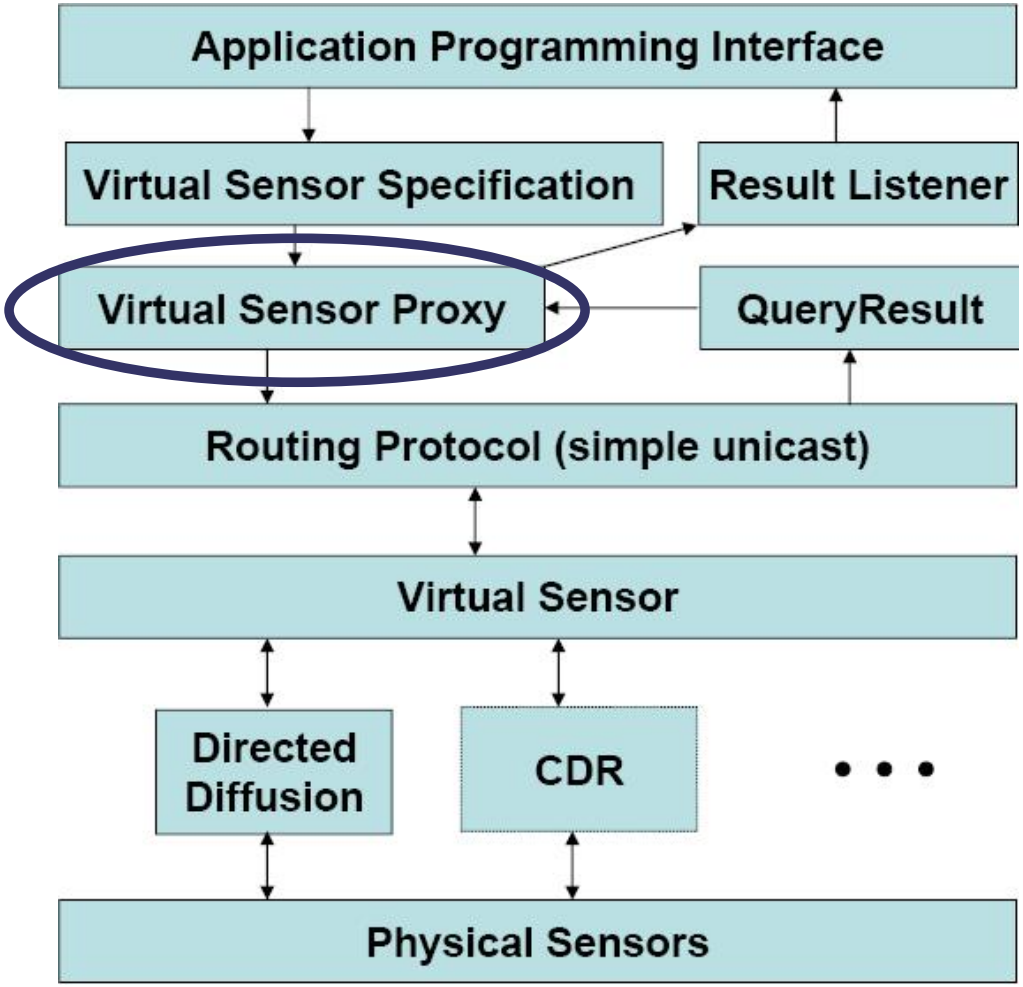
Virtual Sensor Example



Remote handle to listener that receives query result(s)

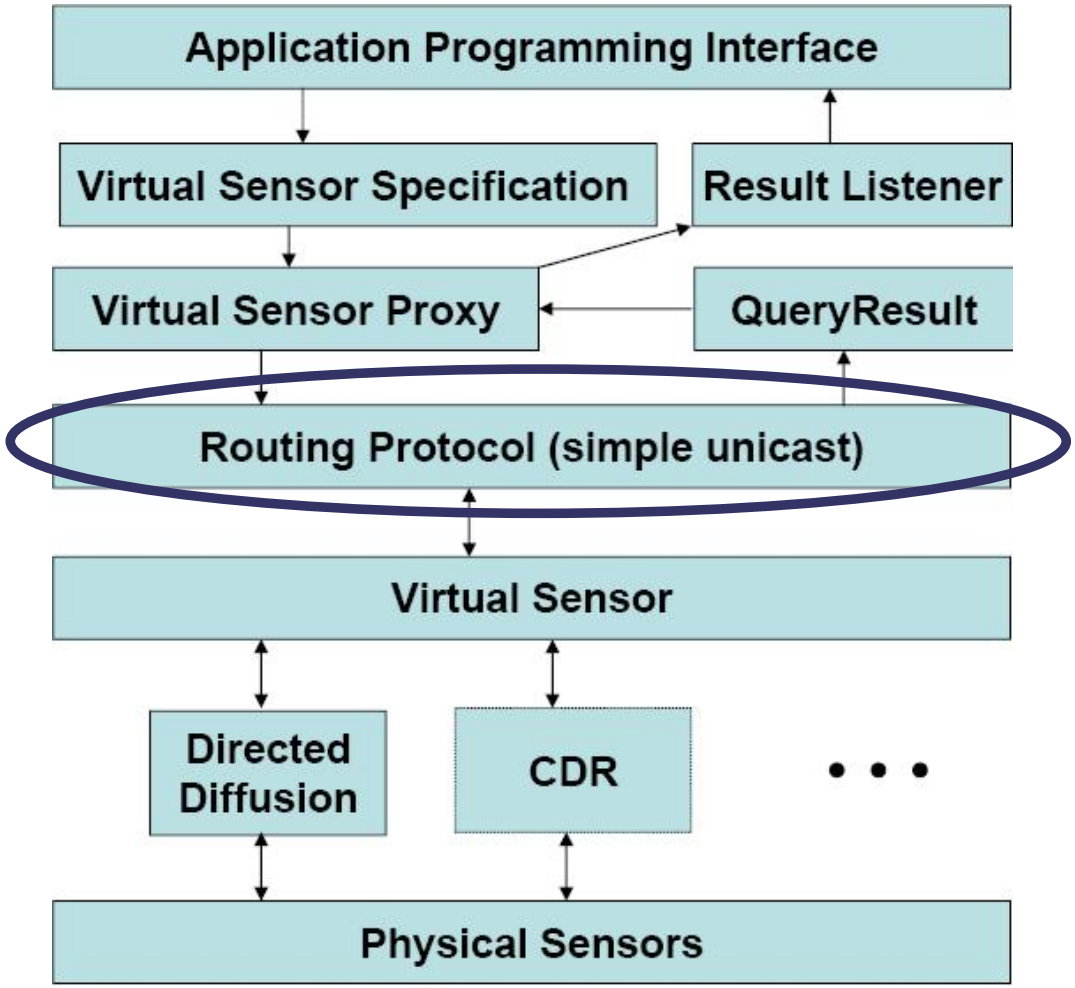
Virtual Sensor Example

- Middleware creates local proxy for virtual sensor
- Keeps a list of live queries and listeners
  - Support multiple applications

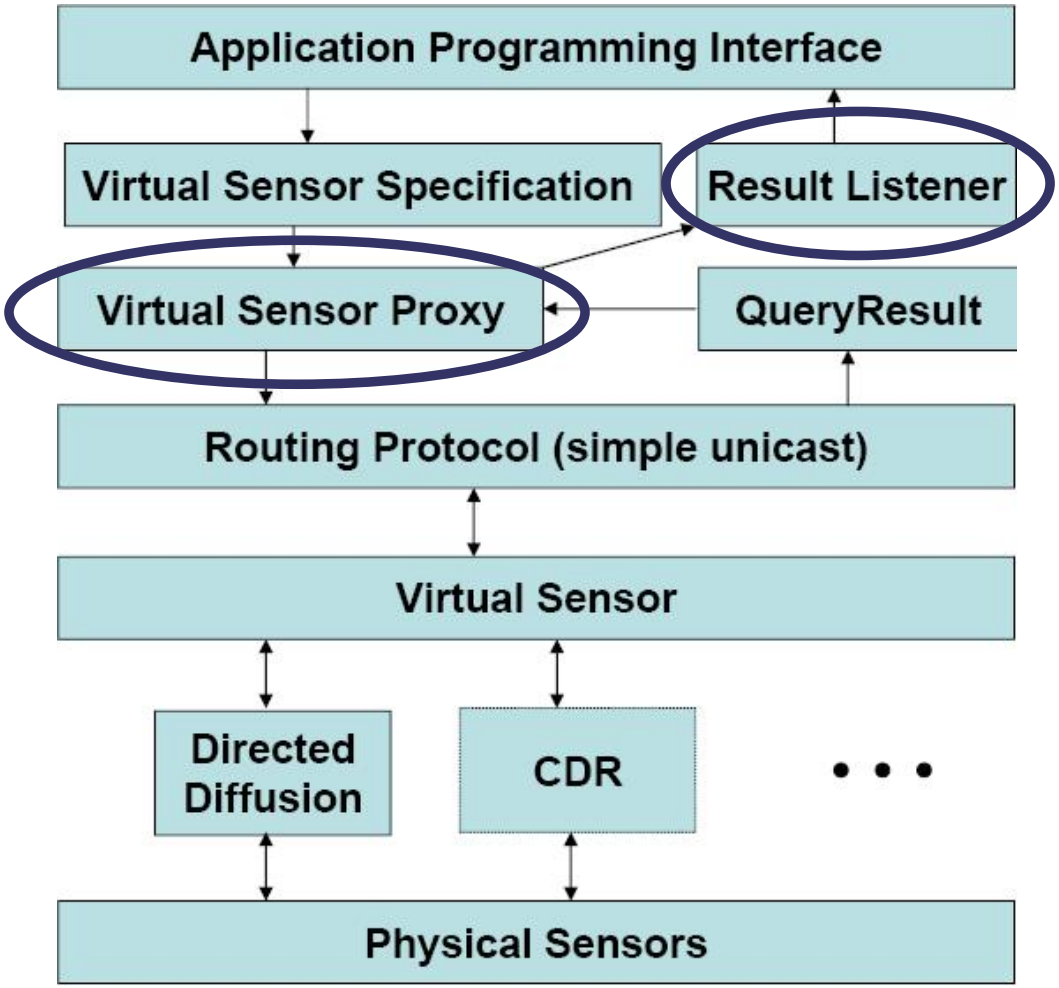


Virtual Sensor Example

Proxy uses unicast routing protocol to connect to remote virtual sensor and collect desired information

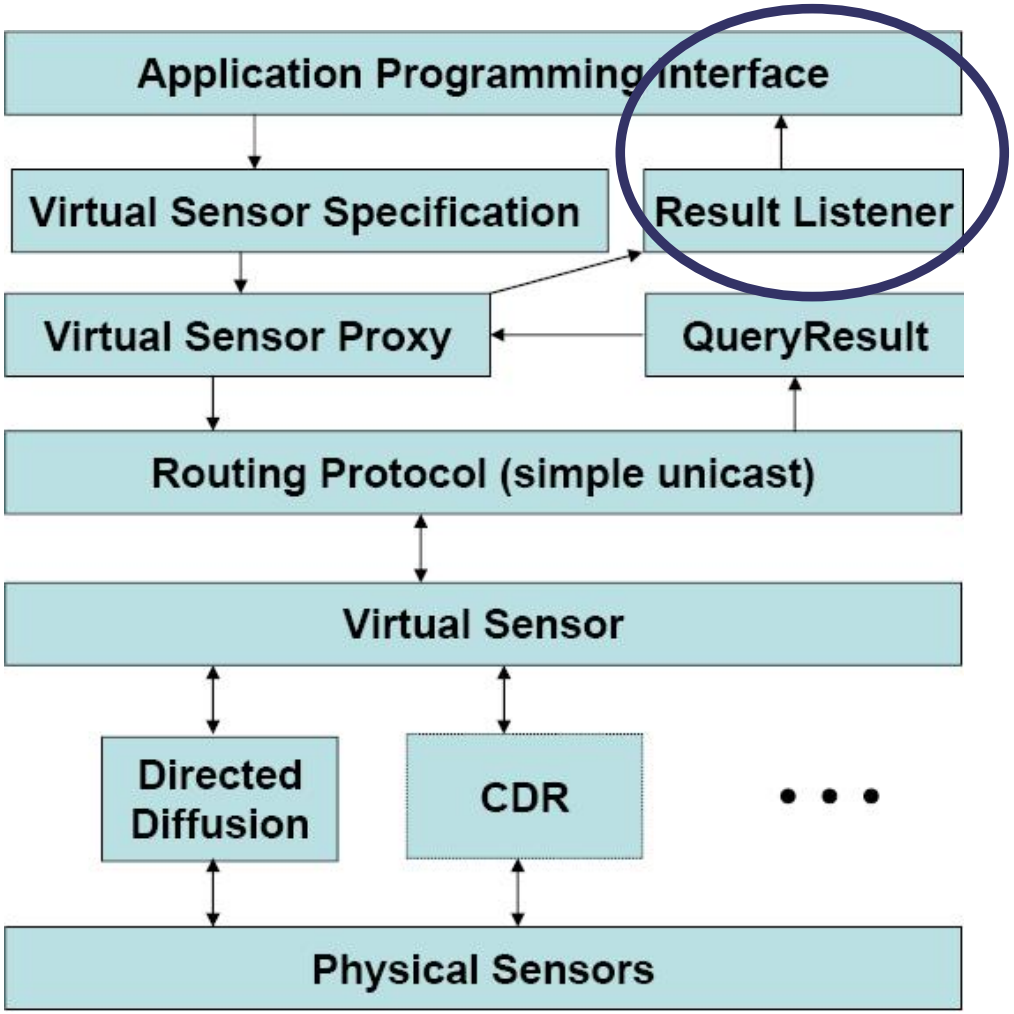


Virtual Sensor Example



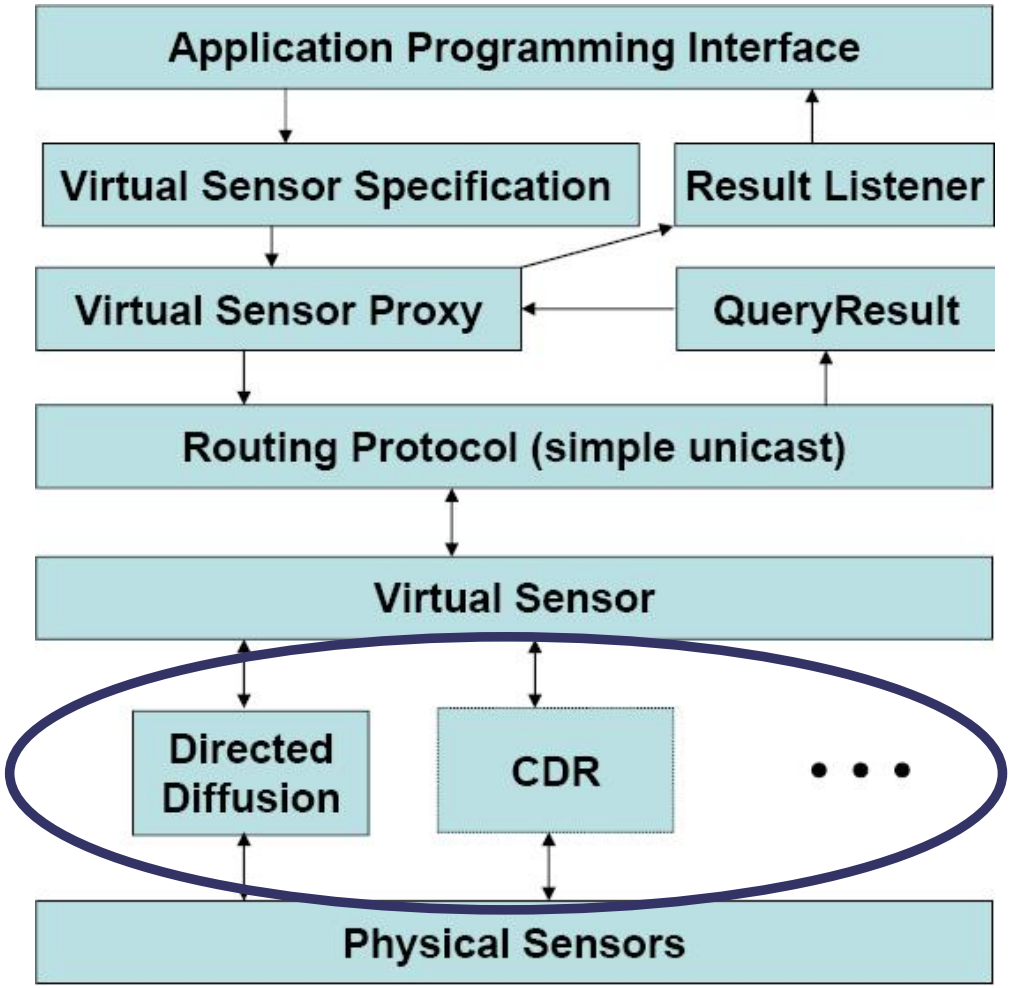
When the result is ready, proxy makes a callback to user's result listener once (for one-time) or periodically (for persistent)

Virtual Sensor Example



Middleware calls `resultReceived(QueryResult)` method for result listeners to forward results to application

Virtual Sensor Example



What about physical sensor discovery?

Physical Sensor Discovery (1)

- Application delegates discovery of physical sensors to middleware that locates actual (physical) sensors based on specified input data types
- Existing communication implementations can be used for sensor discovery and communication:
  - *Directed diffusion*: Virtual sensor propagates an interest message for each data type it requires, creating gradients for funneling information back to the virtual sensor
  - *CDR*: Similar process, but once a source is selected, unicast alone used to communicate with it

Physical Sensor Discovery (2)

- Different protocol implementations providing the necessary data-centric interface can be swapped into middleware:
  - Communication protocol used by the middleware initially broadcasts a data type requirement across local sensor network
  - Sensors that can provide that data type respond
  - Least latency, shortest path, etc. can be used to select a particular data source from many
  - Selection can be refreshed if data source selected becomes unavailable
- Middleware uses (consistency) frequency, aggfreq, to determine:
  - how often sensor selections need to be refreshed
  - whether or not to use CDR, directed diffusion, or some other available protocol

Conclusions

- Virtual sensors
  - allow measurements of abstract data types
  - abstract a set of physical sensors and operations that are performed on them, providing a new way of extracting data from heterogeneous wireless sensors
  - allow a programmer to describe the behavior of a virtual sensor without having to specify underlying details of how it should be constructed
  - offer a way to tailor a generic sensing environment to specific applications
- Future work
  - “Mobile” virtual sensors: Virtual sensors that can move with an event of interest or in response to user movement

Questions?

Thank you!

http://mpc.ece.utexas.edu