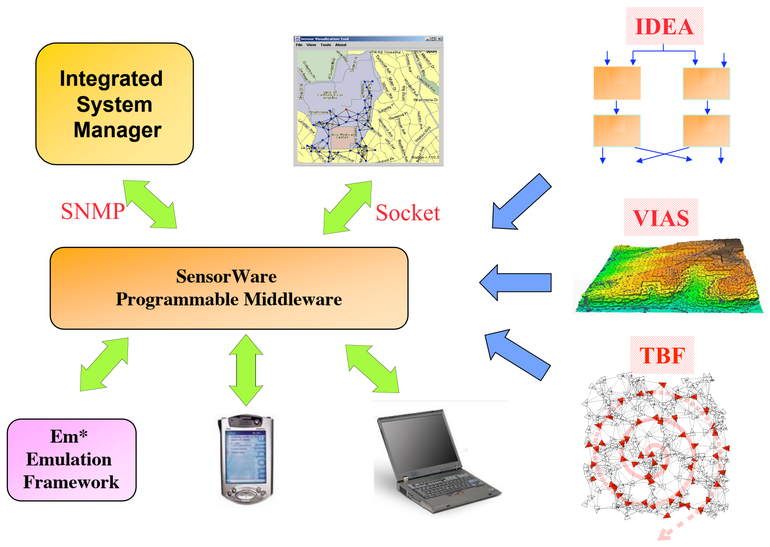
Center for Embedded Networked Sensing

SensorWare in Use

Chih-Chieh Han, Athanassios Boulis, Roy Shea, and Mani Srivastava
Networked and Embedded Systems Lab (NESL) – http://nesl.ee.ucla.edu

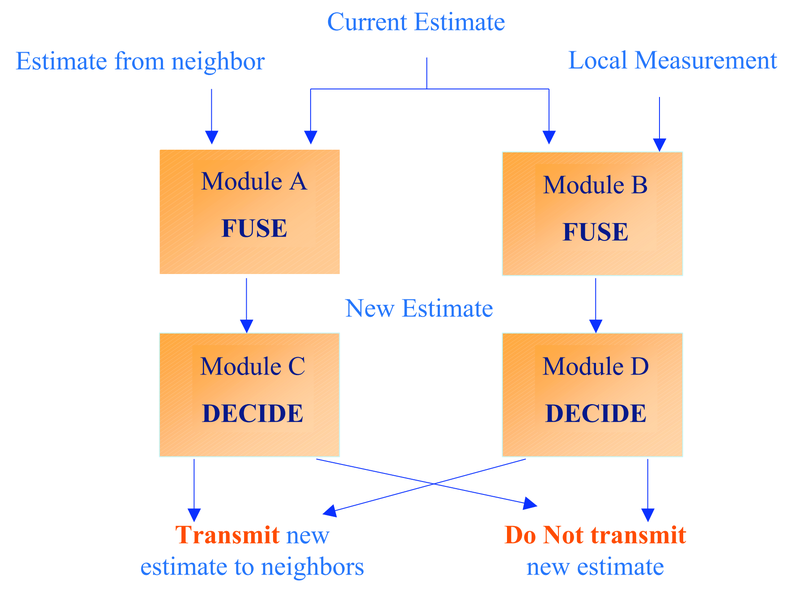
SensorWare: Flexible Extensible System Architecture



Software Features

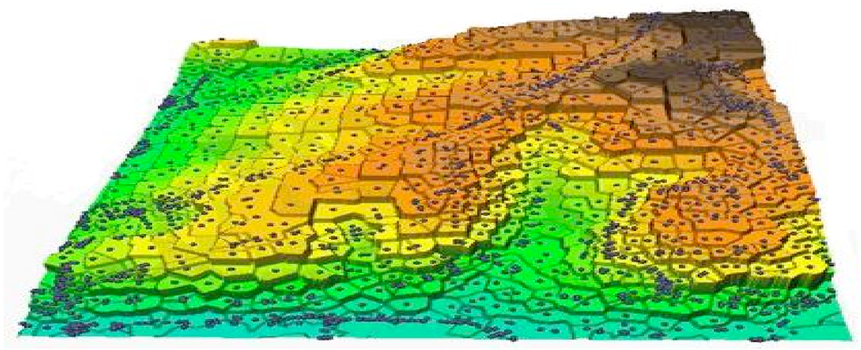
- Dynamic device registration through scripts
- Script code compression
- Script caching and selective script population
- Node Addressing tied with Routing
- State carrying implemented as parameter passing
- Higher-level services available (GPSR, neighbor discovery)
- Multi-routing multi-radio protocol stack

IDEA: Intrinsically Distributed Estimation Algorithm



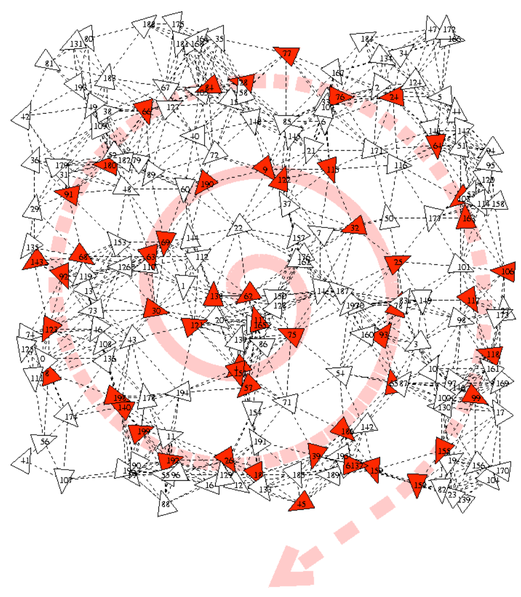

```
parameter span sample_period threshold
replicate neighbor 0                          <- Populate the whole network through controlled flooding
initializations:
  set estim "0 1000"
  interest light l_estimate $sample_period    <- Descriptions of 2 events
  interest timer lifetime $span;
while (1){
main loop:
  wait packet l_estimate lifetime             <- Wait on 3 possible events
  if {$event_name==lifetime} {break}
  if {$event_name== packet} {
    update $estim_table $event_body
    fuse_global $estim $event_body }
  else { fuse_local $estim $event_body }
  if { [check $threshold $estim $estim_table]==1 } {
    send neighbor 0 $estim} }
```

VIAS: Voronoi Interpolated Aggregation Service



```
parameter level
incr level                                    <- Recording the level of aggregation tree
replicate neighbor 0
initializations:
  set weight_sum [expr "[query light] * $weight"]
  set weight [query voronoi]
  interest timer t [expr "3000 - $level * 200"]   <- Set timer according to the level of aggregation tree
while (1){
main loop:
  wait packet t
  if {$event_name == packet} {
    aggregate weight_sum weight $event_body }
  if {$event_name == t} {
    send neighbor $parent_node $parent_mbox "$weight $weight_sum"   <- Send aggregated result to parent node
    break } }
```

TBF: Trajectory Based Forwarding



```
parameter x_func y_func t final_cond inc
initializations:
  set nei [query neighbor]
  set loc [query location]
  set t [expr "$t + $inc"]
  if {[expr $final_cond]} { exit }
  set t_best $t
  set target "[expr $x_func] [expr $y_func]"
  set min_dist [dist $loc $target]
  set min_node [id -n]

while (1){
route finding:
  foreach i $nei {
    set d [dist [query location $i] $target]
    if {$d < $min_dist} {
      set min_dist $d
      set min_node $i
      set t_best $t }
  }
  set t [expr "$t + $inc"]
  if {[expr $final_cond]} { break }
  set target "[expr $x_func] [expr $y_func]" }
if {$min_node == [id -n]} { exit }
set t $t_best
migrate neighbor $min_node                    <- Complete state transfer to next hop
```

UCLA – UCR – Caltech – USC – CSU – JPL – UC Merced