



# Hazard Avoidance in Wireless Sensor and Actor Networks

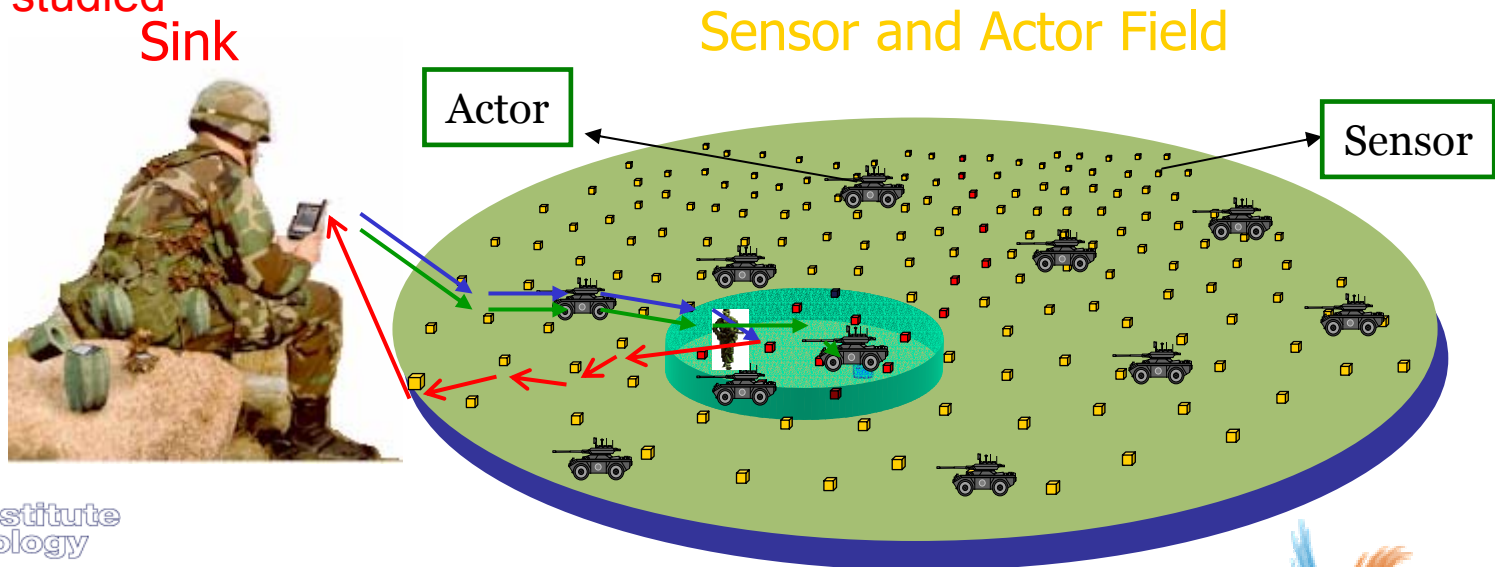
---

Ramanuja Vedantham  
Zhenyun Zhuang  
Prof. Raghupathy Sivakumar

GNAN Research Group  
Georgia Institute of Technology  
<http://www.ece.gatech.edu/research/GNAN>

# Context

- Wireless Sensor Network (WSN)
  - Sink : sends queries and collects data from sensors
  - Sensor : monitors phenomenon and reports to sink
- What next?
  - If there are devices capable of acting on the environment, sink could issue a command
- Traditionally, problems in WSNs have been addressed [Intanagonwiwat'00,Heinzelman'00,Sankarasubramaniam'03,Park'04]
- ☞ Surprisingly, problems in Wireless Sensor and Actor Networks (WSANs) have not been studied

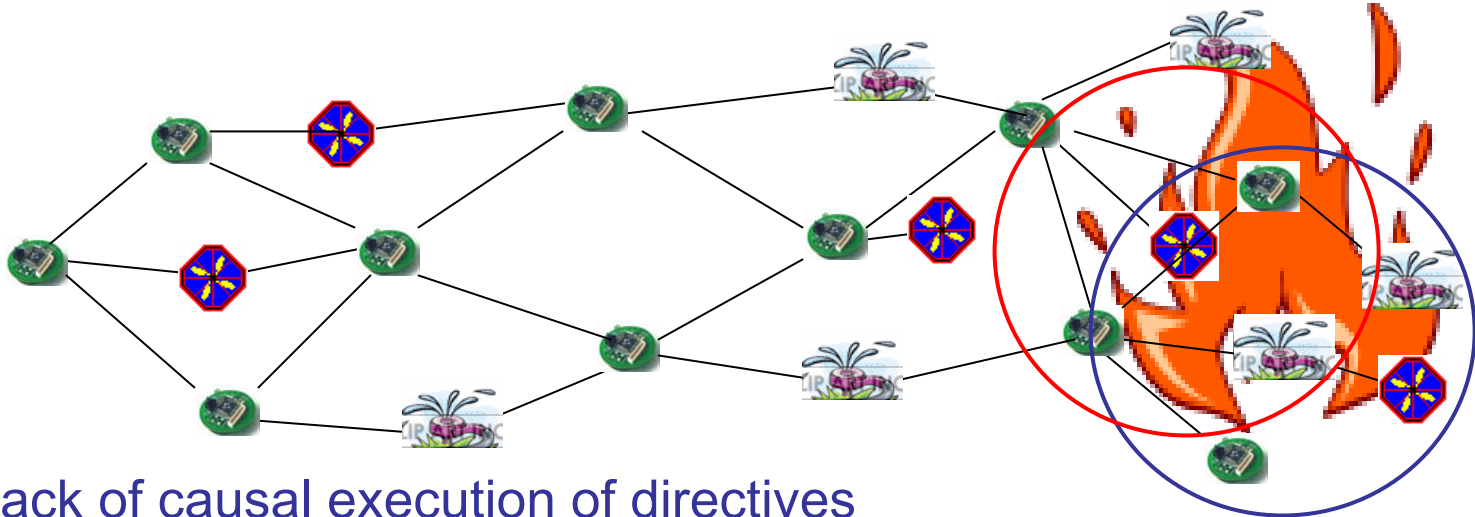


# Characteristics of WSNs

---

- Static, multi-hop wireless network
  - **Sink:** Collects data from sensors and sends queries and commands
  - **Sensors:** Monitors phenomenon and report it to sink
  - **Actors:** Acts on the environment based on the command issued by the sink
  - Example: Environment monitoring
- Sensors
  - Limited battery capacity ( $< 1\text{J}$ ), small memory ( $< 1\text{MB}$ ) and processing power, large number of nodes, high density of nodes [Park'04]
- Actors
  - Larger battery capacity, larger memory and processing power, small number of nodes, low density of nodes [Akyildiz'04]
- Network
  - Low bandwidth ( $< 100\text{Kbps}$ ), frequent node failures [Akyildiz'04]

# Hazards



- Hazards: Lack of causal execution of directives
  - Out-of-order execution of directives from the order that the sink has issued, due to lack of coordination between sensors, actors and the sink
  - Leads to potentially undesirable changes in the environment
- Reason for hazards
  - Different latencies
    - For different sensors and actors randomly located in an event region, the paths differ, and hence, the latencies may differ
    - For a single sensor or actor, different directives may have different latencies



# Goals

---

- **Correctness**
  - Ensure 100% hazard-free operation
  - Causal execution of directives
- **Throughput**
  - Maximize the rate at which queries and commands are processed
  - Define a parameter, directives execution throughput, and maximize it
  - Control loop delay
- **Overhead (subject to above two goals)**
  - Reduce the overall traffic generated while addressing hazards
  - Reduce the resource and energy consumption at sensors and actors

# Observations on Hazards

- Generic hazard avoidance goal
  - For any two entities  $D_x, D_y$  with overlapping execution ranges  $A(D_x)$  and  $A(D_y)$ , and any two sequential instructions,  $I_j \rightarrow I_i$

■ Dependency region: Minimum region surrounding an entity, where hazard-free operation is required

$$t_{I_j \cdot D_x} > t_{I_i \cdot D_x}$$

$$t_{I_j \cdot D_x} > t_{I_i \cdot D_y}$$

$$t_{I_j \cdot D_y} > t_{I_i \cdot D_x}$$

$$t_{I_j \cdot D_y} > t_{I_i \cdot D_y}$$

- Observations for hazard-free operation

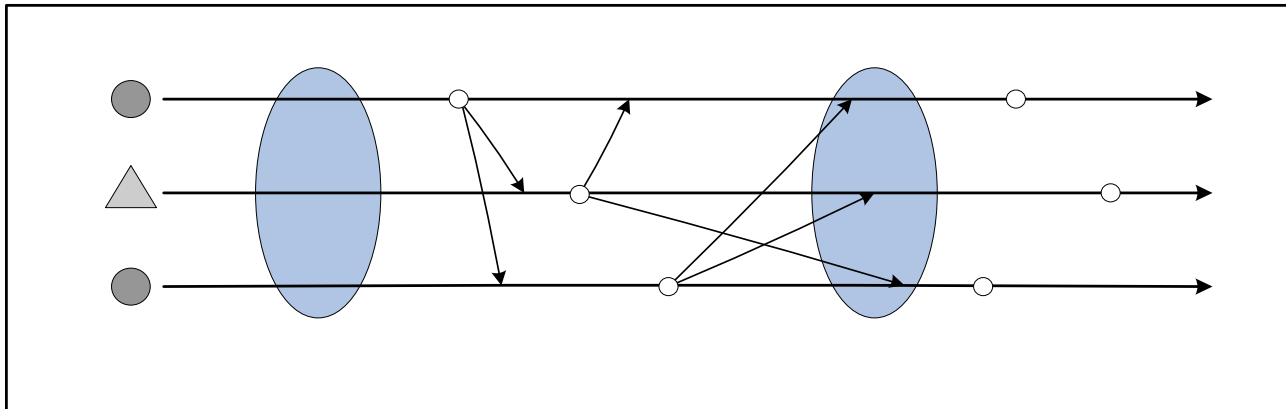
- ☞ Any pair of dependent directives issued to entities that do not have any overlapping execution regions can be executed concurrently across the two entities, although the relative ordering must be preserved within each entity
- ☞ Any pair of dependent directives issued to entities with overlapping execution regions needs to be ordered in the union of the two regions

# The Neighborhood Clock

- Every sensor and actor in the event region has a virtual, local clock
- Local clock is used to provide virtual clock synchronization among all nodes within a dependency region
  - ☞ Ensure that the local clock values are the same within a dependency region
- ☞ Neighborhood Clock (NC) introduces the notion of *a single, virtual clock among all nodes* within every *dependency* region
- ☞ Execution of directives based on the neighborhood clock
- There is a dependency region associated with each entity in the event region
- For this reason, we refer to the local clock itself as neighborhood clock in the NC approach

# Neighborhood Clock Operation

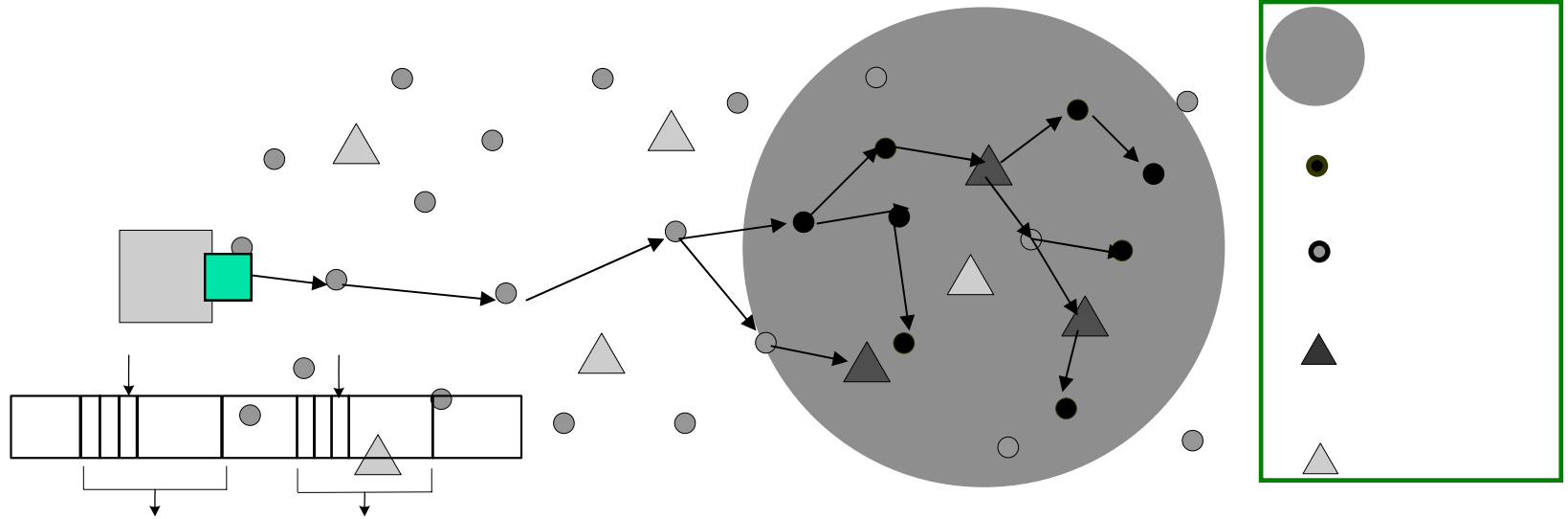
- Virtual neighborhood clock on every sensor and actor for ordering within every *dependency* region
  - Sink creates a unique reference clock initially and sensors and actors initialize their clocks to this value
  - Each entity,  $D_x$ , maintains its own *view* of the progress in the network, based on its neighborhood clock identifier,  $NC(x)$ , where the view number is set to be  $NC(x) + 1$
  - Each sensor and actor will move to the next view only after all other sensors and actors have moved into its current view
- ☞ A directive is executed only when the NC views within a dependency region are synchronized and when they have executed all prior directives



# Why NC?

- 100% hazard avoidance
  - Consider any two entities  $D_x, D_y$  with overlapping execution ranges  $A(D_x)$  and  $A(D_y)$ , and any two sequential instructions,  $I_j \rightarrow I_i$
  - For a generic XAY hazard to happen
    - $D_x$  executes  $I_j$  prior to  $I_i$  by  $D_y$
    - This implies that  $D_x$  is in view  $I_{j+1}$  and  $D_y$  is in view  $I_j$
    - But this is impossible because  $D_x$  will wait for views to be synchronized within dependency region before executing directive
  - ☞ No hazards
- Dependency region-based
  - Smallest region within which hazards need to be addressed
  - ☞ Minimizes delay and maximizes throughput
  - Reduces overhead for hazard avoidance as only views of nodes within dependency region need to be synchronized
- Virtual clock
  - Event-driven approach
  - ☞ No fine-grained timers

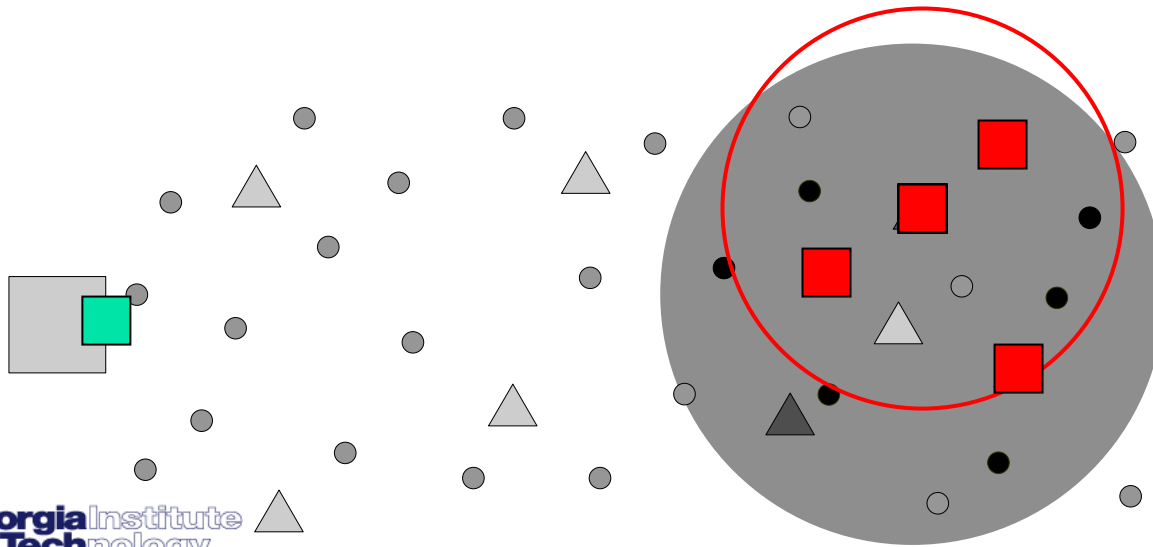
# Distributed NC (1/2)



- Operations at sink:
  - When an event has been detected, the sink creates *reference clock*
  - The sink issues the *START()* control message to all sensors and actors in the event region with the reference clock information
  - When a sensor or actor receives the information, it sets the initial value of the NC clock to the reference clock

# Distributed NC(2/2)

- Operations at sensors/actors: When a node,  $m$ , receives the  $k^{th}$  directive, the following operations are performed:
  - If the clock piggybacked with the directive requires synchronization with  $NC_m(i)$  and  $RC(k) = NC_m(i)+1$ , then action is performed and  $ACK()$  message is sent to neighbors
  - If the neighborhood clock,  $NC_m(i)$ , is at least 2 less than the reference clock is queued and the action is performed later
  - If an acknowledgement,  $ACK()$ , is received:
    - The node identifier is first added to the neighbor list from which  $ACK()$  has been received for this directive
    - If  $ACK()$  from all nodes have been received, the NC value is incremented and the next queued directive is executed



$$RC(k) = NC_m(i)+1$$

$$RC(k) > NC_m(i)+1$$

$ACK(j)$

- Add  $j$  to Nbh\_List

# Other Extensions

---

- Applications where there is a single event and complete ordering among directives is necessary
  - Applications may have different types of events and need not require complete ordering
  - Challenges
    - Parallel dependency among directives
    - Opportunistic execution of directives
    - Merging of events
    - Pruning of events
    - Selective directives
- ➡ Vector Neighborhood Clock and other extensions to the NC approach to address these challenges

# Performance: Baseline Approaches

- Bounded Delay (BD) approach:
  - Sink waits for a specified amount of time after issuing a query or command
  - After issuing a query, sink waits for a time  $T_{Ws}$  corresponding to a fixed specified time to receive all responses from sensors
  - After issuing a command, sink waits for a time  $T_{Wa}$  corresponding to an estimate fixed time before which the command is executed
- Wait-For-All (WFA) approach:
  - Sink issues next directive only after it receives all responses or notifications of execution of that directive from all entities in the event region
  - If the sink sent a command, it will wait for all notifications about the completion of the command from all actors before it issues another query or command

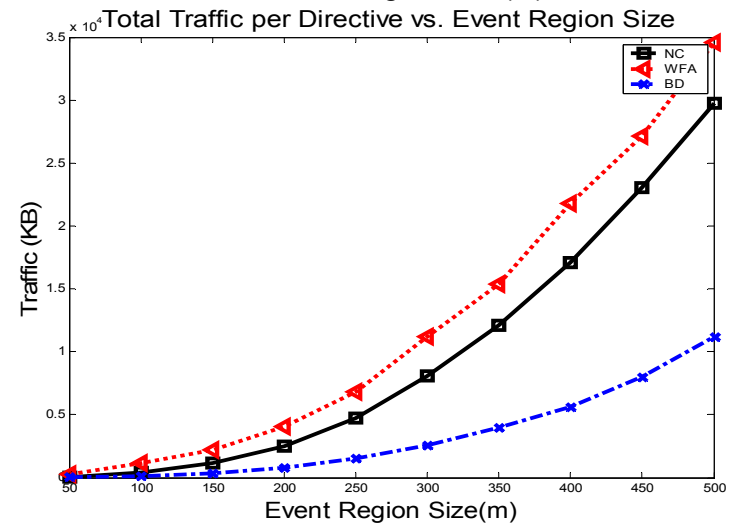
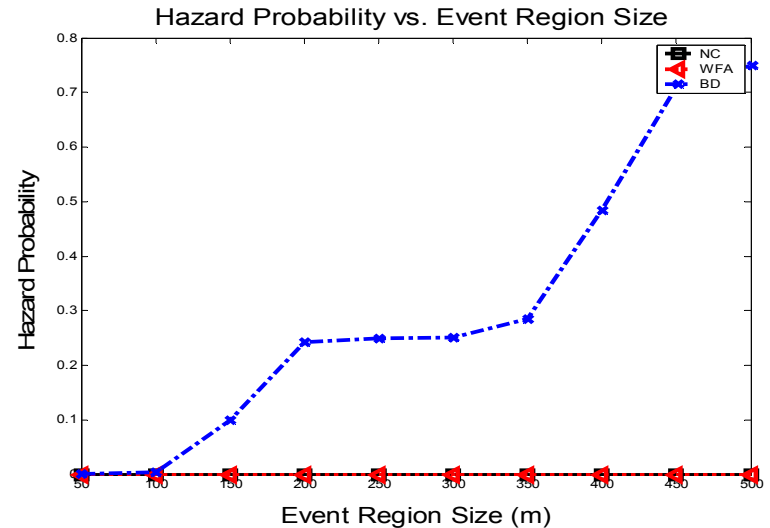
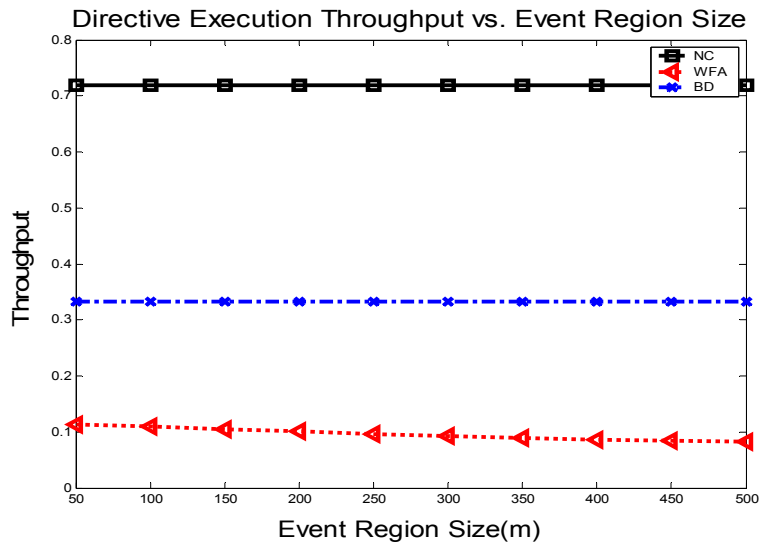
# Performance: Simulation Environment

---

- Simulation environment:
  - Event driven network simulator
  - 1000 nodes in 2000m \* 2000m square area
  - Sensing, acting and communication range = 30m
  - $T_{EP} = 2\text{sec}$ ,  $T_{BD}(\text{query}) = T_{EP}$ ,  $T_{BD}(\text{command}) = 2 * T_{EP}$
  - 100 directives of 1KB each, 50% query, 50% command
- Metrics
  - Correctness (hazard probability%)
  - Directives execution throughput (directives/sec)
  - Total Traffic (KB)

# Performance: Event Area Size

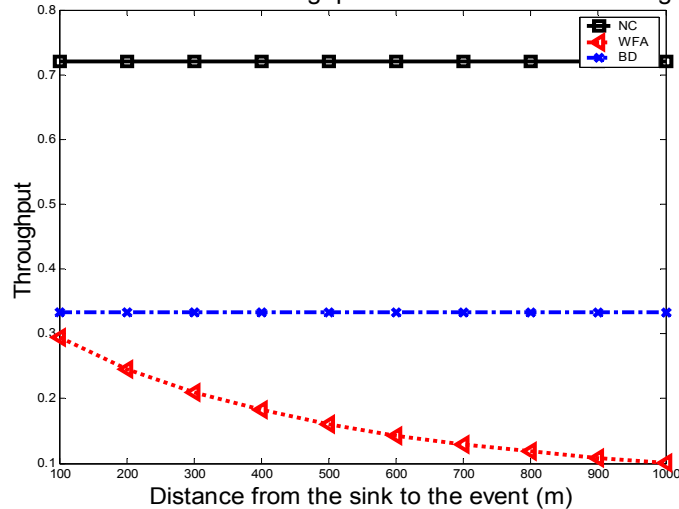
- ✓ BD does not guarantee 100% correctness. NC and WFA have 100% correctness
- ✓ NC approach has the highest directives execution throughput
- ✓ NC approach has slightly lesser traffic overhead than WFA



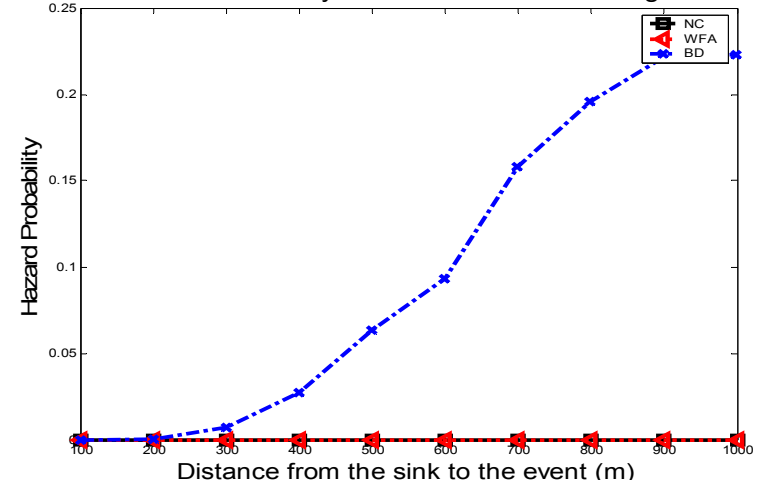
# Performance: Event Distance to Sink

- ✓ BD does not guarantee 100% correctness. NC and WFA have 100% correctness
- ✓ NC approach has the highest directives execution throughput
- ✓ NC approach has lesser traffic overhead beyond 500m

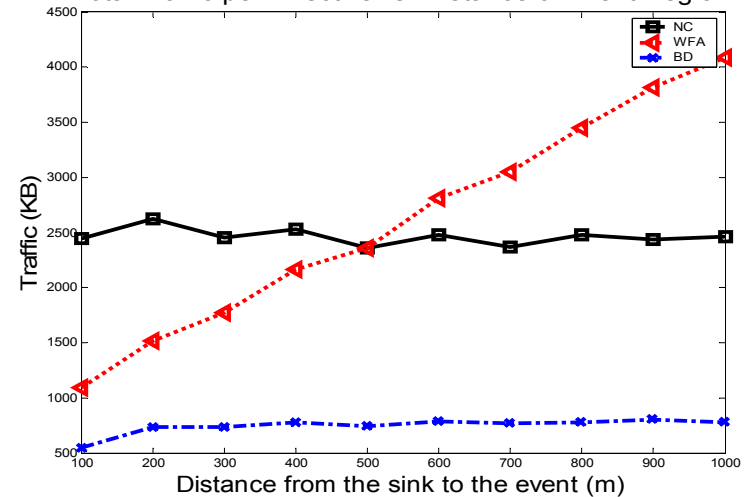
Directive Execution Throughput vs. Distance of Event Region



Hazard Probability vs. Distance of Event Region



Total Traffic per Directive vs. Distance of Event Region



# Related Work

---

- Data hazards when instructions are pipelined in a computer architecture environment [**Hennessy'02**]
- Synchronization problem in a parallel programming environment [**Silberschatz'01**]
- Cooperation in multi-processor environment in distributed systems [**Coulouris'01**]
- ☞ Based on software primitives and hardware techniques and do not address hazards in the context of a WSAN environment
- ☞ Do not have a notion of dependency region
- ☞ Do not address the associated challenges in this environment

# Conclusions

---

- Causal execution of directives (Hazard Avoidance)
  - Identification of different types of hazards in WSNs
  - Identification of the associated challenges and goals in a WSN environment
  - Identification of design principles needed to address the problem
  - Design of a distributed approach to address the problem and the challenges
  - Evaluation of the proposed approach with two baseline approaches