Scalable and Efficient Pattern Recognition Classifier for WSN

Nomica Imran and Asad I. Khan

School of information Technology,
Monash University, Clayton, Victoria, Australia
{nomicac,asad.khan}@infotech.monash.edu.au

Abstract. We present a light-weight event classification scheme, called Identifier based Graph Neuron (IGN). This scheme is based on highly distributed associative memory. The local state of an event is recognize through *locally* assigned identifiers. These nodes run an iterative algorithm to coordinate with other nodes to reach a consensus about the global state of the event. The proposed approach not only conserves the power resources of sensor nodes but is also effectively scalable to large scale WSNs.

1 System Model

In this section, we model the wireless sensor network. Let there are N sensor nodes in the wireless sensor network. Let $\mathcal{E} = \{e_1, e_2, \ldots, e_E\}$ be a non-empty finite set of such data elements sensors sense from their surroundings. Input data is in the form of *patterns*. A pattern over $\mathcal{E}$ is a finite sequence of elements from $\mathcal{E}$. The length of a pattern is the number of sensors in the system. We define $\mathcal{P}$ as a set of all possible patterns of length L over $\mathcal{E}$. We model GN as a structured overlay $\mathcal{G} = \{(\mathcal{E} \times \mathcal{L})\}$ where $\mathcal{L} = \{1, 2, \ldots, L\}$, where $n(e_i, j)$ is a node in $\mathcal{G}$ at i-th row and j-th column. GN can be visualized as a two dimensional array of L rows and E columns. Total number of nodes in the $\mathcal{G}$ are $E \times L$. We refer all the nodes in the $(j-1)$ column as the left neighbors of any node $n(*, j)$ in j-th column. Similarly, all the nodes in the $(j+1)$ column are called as the right neighbors of $n(*, j)$. In a GN-based classifier[Raja,2008],[Nasution,2008], each node $n(e_i, j)$ is programmed to respond to only a specific element e_i at a particular position j in a pattern. That is, node $n(e_i, j)$ can only process all those patterns in $\mathcal{P}$ such that e_i is at the j-th position in that pattern. Each node maintains an *active/inactive* state flag to identify whether it is processing the incoming pattern or not. Initially all nodes are in inactive state. Upon arrival of a pattern, if a node finds its programmed element e_i at the given position in the pattern, it switches its state to active otherwise it remains inactive. Only active nodes participate in our algorithm and inactive nodes remain idle. At any time, there can be exactly L active nodes in a GN. Hence, there are exactly one active left-neighbor and exactly one active right-neighbor of a node $n(e_i, j)$ where $j \neq 0, l$. Whereas terminal nodes $n(e_i, 0)$ and $n(e_i, L)$ has only one active left and right neighbor respectively.

P. Sénac, M. Ott, and A. Seneviratne (Eds.): MobiQuitous 2010, LNICST 73, pp. 303–304, 2012.

2 Proposed Protocol

On arrival of an input pattern $\mathcal{P}$, each active node $n(e_i, j)$ store e_i in its j_{th} position. Each node $n(e_i, j)$ sends its matched element e_i to its active neighbors $(j + 1)$ and $(j - 1)$. The GNs at the edges will send there matched elements to there penultimate neigbours only. Upon receipt of the message, the active neighboring nodes update there bais array. Each active node $n(e_i, j)$ will assign a local state L_s to the received $(e_i,)$ value. The generated local state L_s will be *Recall* if the the added value is already present in the bais array of the active node and it will be a store if in-case its new value. An $< ID >$ will be generated against each added value. The rules for assigning ids are as under:

- Rule 1. $Store_{(Si)} > Recall_{(Ri)}$: If node $n(e_i, j)$ self local state L_s is *Recall* but it receives a *Store* command from any of its neighbors, $(j+1)$ or $(j-1)$, it will update its own state from $Recall_{(Ri)}$ to $Store_{(Si)}$.
- Rule 2. All New Elements: If any of the elements presented to $\mathcal{G}$ is not existing in the bais array of any of the active nodes $n(e_i, j)$ suggests that its a new pattern. Each active node $n(e_i, j)$ will create a new <ID> by incrementing the already stored maximum <ID> in there bias array by 1.
- Rule 3. All Recalls with Same ID: If e_i presented to $\mathcal{G}$ is the recall of previously stored pattern with same <ID>, means that its a repetitive pattern. The same <ID> will be allocated to this pattern.
- Rule 4. All Recalls with Different IDs: If all the e_i of the pattern $\mathcal{P}$ presented to $\mathcal{G}$ are the recall of already stored patterns with different <IDs> indicates that it's a new pattern. Each active node $n(e_i, j)$ will find out the $max(ID)$ in there bias array and will increment it by 1.
- Rule 5. Mix of Store and Recall: If the end decision is to *Store* due to mix of *Store* and *Recall*, each active node $n(e_i, j)$ will again find out the $max(ID)$ in there bais array and will increament it by 1.
- Transition Rule 1 If the active node $n(e_i, j)$ has received a greater value from its left neighbor $(j + 1)$, it will upgrade its local state and transfer this updated value to its right $(j - 1)$ neighbor only.
- Transition Rule 2 In case if the received value from both the neighbors $(j+1)$ and $(j-1)$ are smaller than the local value, node $n(e_i, j)$will upgrade its value and transfer this new value to both of its neighbors. Once the pattern has been stored, a signal is sent out within the network informing all the IGN nodes that the pattern has been stored. This is called the pattern resolution phase.

3 Conclusion and Future Work

In this paper we have proposed an in-network based pattern matching approach for providing scalable and energy efficient pattern recognition within a sensor network. Our proposed scheme is base station independent. The proposed scheme is also independent of preprocessing steps such as patterns segmentation or training for its processing. Through parallel processing, the scalability issues in WSN are catered well.