Content-based Clustering in Flooding-based Routing: The case of Decentralized Control Systems

Soroush Afkhami Meybodi, Jan Bendtsen, Jens Dalsgaard Nielsen
Department of Electronic Systems
Aalborg University
Aalborg, Denmark
Emails: {sam, dimon, jdn}@es.aau.dk

***Abstract*—This paper investigates a problem that is usually studied in communication theory, namely *routing* in wireless networks, but it offers a control oriented solution – particularly for decentralized control systems – by introducing a new routing metric. Routing algorithms in wireless networks have a strong impact on the performance of networked control systems which are built upon them, by imposing latency, jitter, and packet drop out. Here, we have gone one step further from only investigating the effect of such communication constraints, and have directly intervened in the design of the routing algorithm for control systems in order to: 1) realize our preferred network topology and data traffic pattern, and 2) making it feasible to add and remove sensors, actuators, and controllers without having to decommission and/or re-design the system. Moreover, the end-to-end latency and jitter in our system tend to be minimal as a result of robustness of the algorithm to topology modifications. The proposed routing solution combines traditional flooding-based routing scheme with a novel method of clustering nodes based on correlation analysis between existing and emergent sensors and actuators of a control system.**

***Keywords*-routing; flooding; clustering; system identification**

I. Introduction

Chronologically, flooding is the first type of routing solutions that appeared [1]. The name describes well how it works. In pure flooding – that is flooding without any network structure – when a node receives a packet, it checks whether it is the final recipient of the packet or it has received the same packet before. The latter may happen if a packet reaches the same node from different routes. A negative answer to both questions results in retransmission of the packet. Assuming no congestion at the MAC layer, flooding is unquestionably the fastest routing method with the minimum latency which offers a true peer-to-peer (P2P) traffic support. It is not sensitive to modifications of the physical topology because it does not rely on identifying and keeping track of optimal routes. Actually, no topology maintenance is required at all.

All of these advantages come at the expense of a serious drawback. Flooding imposes a heavy overhead, hence is painfully resource consuming. Too many healthy retransmissions happen before a packet is faded from the network. This will not only consume much energy, which is especially valuable for battery-operated nodes, but also causes congestion and increases collision chance at the MAC layer. This diminishes the main benefits of flooding and increases energy consumption both because of the high number of transmissions and the high number of re-transmissions after collisions at the MAC layer happen. That is why pure flooding works well only for small networks with a few number of nodes [1].

Several remedies have been proposed to reduce the number of unnecessary retransmissions in flooding-based routing. Some assume that a limited number of retransmissions are enough to reach the destination and do not propagate the packet any further. Such a number is derived either probabilistically or deterministically by considering the worst case [1]. All of the other methods impose a structure on the network. In the context of flooding-based routing, structuring a network is equivalent to partitioning it into separate or overlapping clusters.

Clustering confines the domain of flooding each packet and can be done by any of the following methods:

- *Coordinates-based Clustering*: Clustering nodes based on their geographical, relative, or virtual coordinates is a simple task provided that the nodes are aware of their geographical location, or can infer their relative or virtual locations. Any kind of distance measure between a node and a cluster center might be used as the membership criterion. If a node is located close enough to the cluster center, it will be a member of that cluster.
- *Metric-based Clustering*: Communication metrics might also be exploited in setting a structure for the network, e.g. by omitting the nodes that have little residual energy or blacklisting the links which are not reliable enough. This category represents a number of popular protocols. Here is how they generally work: At pre-scheduled time intervals, several nodes elect themselves as cluster heads. This could be done by either a pre-defined probability value in each node as in Low-Energy Adaptive Clustering Hierarchy (LEACH) [2], or based on the remaining energy of battery operated nodes as in Hybrid Energy-efficient,

Distributed clustering (HEED) [3]. Then the remaining nodes attach to the *nearest* cluster head. "Near" could be interpreted by any kind of distance measure which is derived from a routing metric, e.g. strength of the received signal, expected transmission count (ETX), hop count, etc. [4].
- *Content-based Clustering*: This is a data-centric approach in which the data content of the packet is used to alleviate flooding overhead. Current solutions are either based on tailoring redundant data or aggregating correlated or similar data [5].

In this paper, we are going to introduce another approach towards content-based clustering which is based on the specific characteristics and requirements of the top layer application, i.e. a decentralized control system. Our solution utilizes control oriented metrics to form clusters [6], instead of typically used communication based metrics. Although clustered flooding-based routing is well known, how to create and maintain these clusters makes our routing solution novel.

The remaining of the paper fulfills the above mentioned objectives by coping with the following structure: In Section II, some preliminaries regarding the traffic pattern of our application, which we will call decentralized Wireless Networked Control Systems (WNCS), are stated. They are followed by introducing the assumed networking topology. The main result is given in Section III by proposing a routing algorithm for decentralized WNCS followed by implementation details and step-by-step procedures on cluster formation and maintenance. Some performance related remarks are presented in Section IV. The paper is concluded in Section V.

II. Preliminaries

A. Dominant Traffic Pattern

In a WNCS, there are three kinds of nodes: actuators, controllers, and sensors. All of them should have the capability to act both as a data *sink* and as a data *source*, described as follows.

- A Sensor is regularly a data source to send sensory data towards relevant controllers, typically once per control *cycle time* interval, equivalent to the control loop sampling time, e.g. 100 ms.
- A sensor sporadically acts as a data sink to receive configuration data from its associated controllers.
- An actuator is regularly a data sink which receives commands from the associated controller at each control cycle time and implements them.
- An actuator sporadically acts as a data source to report failures.
- Controllers should send and receive data in each control cycle time interval. They gather data from sensors at the beginning of a typical cycle time interval, and send commands to the actuators at the end of the interval. This is exactly what happens in a Programmable Logic Controller (PLC), supplied either with single cable inputs/outputs or aggregated bus communication modules.

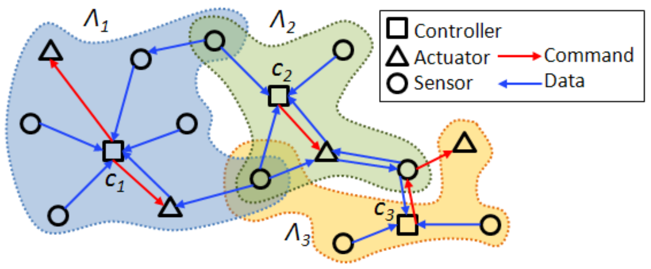

Figure 1. Logical network topology of a simple decentralized wireless networked control system

Unlike newer routing protocols for WNCSs, which assume a Multi-point to Point (MP2P) traffic with controllers as the sink nodes [7], the above list suggests a P2P traffic pattern. Especially the third item, which has the same importance as the first one and happens at the same frequency, makes WNCSs incompatible with MP2P architecture. Note that, this would not be the case if only monitoring and open loop control were of interest as in [7].

B. Network Topology

Fig. 1 shows the logical network topology of a simple decentralized WNCS. Arrows represent data direction in its regular functioning mode, but information flow in the reverse direction is also required as explained earlier in II.*A*.

Fig. 1 depicts the key assumptions that we have considered in topology design, described in the following:

1) There could be a large number of controllers ($C_i, i = 1, ..., n$).
2) Each controller and its associated sensors and actuators form a set, called a *cluster* henceforth. Clusters are identified by their unique tag ($\Lambda_i, i = 1...n$).
3) There are as many clusters as controller nodes which are called *cluster heads*.
4) Each packet contains a cluster *association* field. In general, a packet might be tied to one or more clusters. It is also possible that a packet is not associated with any cluster.
5) A sensor might be a *member* of multiple clusters, meaning that its generated data could be associated to more than one cluster head. In other words, a packet that is generated at a sensor node, might be reported to more than one controller.
6) An actuator could be a member of at most one cluster, meaning that it may not receive commands from more than one controller.
7) Data packets to/from members of a cluster should be sent from/to the cluster head. In other words,

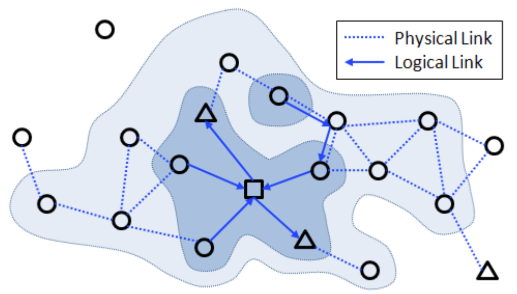

Figure 2. Preserving connectivity of cluster members by a surrounding cloud whose thickness is adjusted by *relay credit*, here defined as hop-count ≤ 2

a controller node is either the source or the final destination in every transmission path.

8) Relaying data packets associated to the cluster (Λ_j) could be done via all of the nodes in the entire network, irrespective of their membership status in Λ_j, provided that the relayed packet has enough remaining *relay credit*.

Relay credit can be defined in terms of any scalar node or link routing metric, e.g. hop count as a node metric or expected transmission count (ETX) as an accumulative link metric [4].

The last assumption clarifies that cluster membership is not necessary when relaying a packet. The purpose of assigning relay credit to a packet is to preserve connectivity in a cluster while constraining the number of retransmissions that might occur to a packet among non-member nodes. Each data packet is given an initial relay credit besides cluster membership tags, when generated. While it is roaming inside its own cluster, it does not spend any relay credit. However, when it is being relayed among non-member nodes, the initial relay credit is decreased at each non-member node until the remaining credit is not enough for more retransmissions among non-member nodes. Fig. 2 illustrates an example.

In Fig. 2, the dark area shows the members of a cluster Λ, and the lighter surrounding area shows the maximum penetration depth of the packets of Λ if hop-count ≤ 2 is considered as the relay credit criterion. It helps preserving connectivity of Λ members, when there is no direct physical link between them. This is shown in Fig. 1 too, where a sensor node which is a member of Λ_3 connects to its own cluster via an actuator and a sensor of Λ_2.

III. The Routing Protocol

A. Control Oriented Clustering of Nodes

To propose a method to form clusters based on requirements of the control system, we rely on the results of [6]. It offers three stochastic correlation-based measures that indicate *usefulness* of incorporating a new sensor/actuator in a present system model. We will use these scalar measures as application-based routing metrics in order to extend the concept of *distance*. Consequently, membership in a specific cluster Λ_j is granted to a node if that node is closer to the cluster head C_j than a specific threshold value d_j.

1) Addition of a new sensor: For a newly added sensor node, the following two complementary measures are proposed.

$$d^2_{U_p,y_a} = \frac{E[y_a(k) - \hat{y}_a(k|U_p^{k-1})]^2}{E[y_a(k) - E(y_a(k))]^2} \tag{1}$$

$$d^2_{U_pY_pY_a,y_a} = \frac{E[y_a(k) - \hat{y}_a(k|U_p^{k-1}, Y_p^{k-1}, Y_a^{k-1})]^2}{E[y_a(k) - E(y_a(k))]^2} \tag{2}$$

in which d^2 represents the correlation based distance and varies between 0 and 1. Subscripts $(.)_p$ and $(.)_a$ refer to present model and added device, respectively. $y(k)$ and $u(k)$ mean individual samples of a sensor's data and an actuator's command at time k, while Y^k and U^k indicate the set of all samples from the beginning up to and including time k. $E(.)$ stands for expected value operator over a finite number of data samples, N, which is pre-defined in the sensor node. Superscript $\hat{(.)}$ stands for the least-squares estimation based on the available model.

With respect to the above mentioned definitions, interpretation of (1) and (2) is given in the following paragraph, assuming that: 1) the model of the present system is discrete-time linear time-invariant and 2) the new sensor provides sufficiently exciting data, and 3) a consistent un-biased least-squares estimation is given when $N \to \infty$.

The denominator in (1) and (2) is the variance of the data gathered by the new sensor, i.e. y_a. The numerator in (1) indicates how predictable the current $y_a(k)$ is if the commands of all actuators are known in the previous samples. If, according to the present model, none of the $k-1$ samples of all of the actuators have any tangible effect on the k^{th} sample of y_a, the following equation holds true.

$$E[y_a(k)|U_p^{k-1}] = E(y_a(k)) \tag{3}$$

Furthermore, if an unbiased estimation is assumed, we have $\hat{y}_a(k|U_p^{k-1}) = E[y_a(k)|U_p^{k-1}]$ which in combination with (3) results in the following expression:

$$\hat{y}_a(k|U_p^{k-1}) = E(y_a(k)) \tag{4}$$

Equation (4) means that the conditional least squares estimation of y_a is equal to its actual expected value. Therefore, the present model is good enough and the new measurement does not add any value to it. In this situation, $d^2_{U_p,y_a} = 1$, which should be read as: the new node is too far from the cluster head and cannot become a member, that is it is irrelevant to the control loop in question.

Equation (1) measures how much the additional sensor is affected by the present actuators in open loop. Nevertheless, this measure only reveals linear correlation. To look for nonlinear correlations, (1) should be modified according to the specific nonlinearity we are looking for. This is the easy step, but the difficult part is to perform nonlinear online incremental system identification to find $\hat{y}_a$. We do not consider this case in this paper.

The numerator in (2) measures how much the additional output could be controlled by the present actuators in closed loop. The interpretation is similar to (1), but this time the data from all of the sensors, including the new one, are also used in the least squares estimation, hence making it a more computationally intensive problem. Either (1) or (2) could be used in a given setting. Exploiting (1) is recommended in cases where y_a cannot be controlled independently of y_p [6].

2) Addition of a new actuator: When a new actuator is added, the following measure is proposed.

$$d^2_{U_a y_p | U_p Y_p} = \frac{E[y_p(k) - \hat{y}_p(k|U_p^{k-1}, Y_p^{k-1}, U_a^{k-1})]^2}{E[y_p(k) - \hat{y}_p(k|U_p^{k-1}, Y_p^{k-1})]^2} \quad (5)$$

Equation (5) measure how much influence the additional actuator has on the present sensors in closed loop. If U_a^{k-1} does not have any effect on improving prediction of $\hat{y}_p$, then the prediction errors in the numerator and denominator will look alike and d will get its maximum value ≈ 1 On the other hand, if U_a^{k-1} is useful such that the prediction error in numerator is much less than that in denominator, then we have: $d \to 0$. Equation (5) should be interpreted similar to the previous measures with similar concerns. The same assumptions hold for a consistent estimation of $\hat{y}_p$. In practice, to provide a sufficiently exciting control signal, the actuator has to be driven by an external signal.
Remark 1: Latency in the communication network has a considerable impact on all of the introduced measures. But at the same time, it influences control performance too. Therefore, it is reasonable to consider this effect on evaluating usefulness of adding new sensors and actuators.

B. Network Layer Packet Format

To illustrate details of the protocol, the packet format shown in Fig. 3, is chosen for the Network layer.

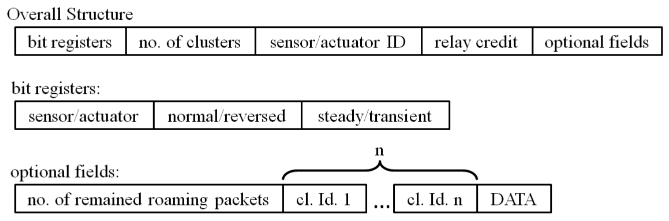

Figure 3. Structure of data packets at Network layer

The first field in the packet is a set of bit registers. The first bit flag determines whether the packet is sensor or actuator related. A sensor packet is generated in one of the sensors and should be routed to one or more controller nodes. An actuator packet is generated in a controller node and should be routed to an actuator node. The second bit indicates the direction of data. For sensor packets, ”normal” means ”from sensor to controller” and ”reverse” means the other way around. The converse is true for actuator packets. The third bit register shows if the packet is sent in *steady* or *transient* operating mode. In transient mode, the packet contains an additional field, namely *number of remained roaming packets*, which is listed among optional fields. The difference between these two modes and the function of roaming packets is explained later in Section III.C.

The second field stores the number of associated clusters. ”Zero” in this field means that the packet is not associated with any cluster. If the packet is linked to $n > 0$ clusters, n additional fields are included in the packet, each of which containing the address of one of the associated clusters.

The next field is *Sensor/Actuator ID*. In sensor packets, it contains address of the sensor node that has generated the packet. In actuator packets, it contains address of the actuator node that the packet is destined to. Assignment of unique addresses to all of the nodes in the entire network is a prerequisite to our routing solution. The same demanding requirement exists in emerging standards, e.g. IETF ROLL, by incorporating IPv6 as a worldwide addressing standard [7].

The packet also contains a *relay credit* ≥ 0 which is explained in details, earlier in Section II.B.

The last optional field is DATA which actually contains the application layer packet.

C. Cluster Formation

Here, we give a high level description of cluster formation. Assume that the controller nodes $(C_i, i = 1, ..., n)$ are deployed as cluster heads. In a realistic scenario, each cluster head has a built-in model of the subsystem it is supposed to control. All of the initially deployed sensor and actuator nodes are already bound to their controllers. In other words, in the network setup phase, all of the nodes are aware of their cluster membership. As a result, sensor nodes immediately start to function in their normal operating mode. See Fig. 4.

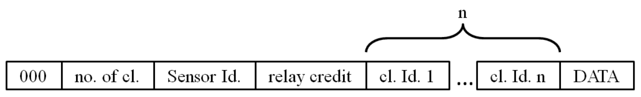

Figure 4. Structure of sensor packets at steady operation

Actuator nodes should start working in a safe mode and wait until they receive commands from the cluster head, i.e. the controller unit. The cluster head starts sending commands to the actuator as soon as it can devise the commands based on received sensor packets and the pre-programmed plant model. Command carrying packets are illustrated in Fig. 5.

In both above cases, the packets flood their pertinent clusters. Moreover, they propagate among the nodes of

100	1	Actuator Id.	relay credit	cl. Id	DATA

Figure 5. Structure of actuator packets at steady operation

neighboring clusters into a certain depth defined by their relay credit.

Later on, when a new sensor pops up, it does not initially belong to any cluster and it is in transient operating mode. Thus, it publishes data in *roaming packets* as shown in Fig. 6.

001	0	Sensor Id.	relay credit	no. of remained roaming packets	DATA

Figure 6. Structure of packets generated by a sensor when it is just turned on

When a roaming packet arrives at a neighboring node that is operating in *Steady* mode, it inherits all of the cluster tags of that node – meaning that the $cl.Id.$ fields of the packet are refreshed. This action is performed only if *relay credit* > 0. If either the neighboring node is in *Transient* mode or *relay credit* $= 0$, cluster tags of the packet remain unchanged.

In steady mode, embedding a cluster tag into a packet gives it the right to freely flood in that cluster without spending relay credit, but it is not the case in transient mode in which relay credit is constantly spent for every transmission. Therefore, embedding cluster tags into roaming packets does not give them a free pass. On the other hand, running out of relay credit is not the stop criterion when retransmitting a packet in transient mode. It just kills its ability to inherit new cluster tags. At the end, a roaming packet floods into the clusters that it managed to enter before running out of relay credit.

Example 1: Fig. 7 depicts an example when a new sensor is placed amongst nodes of Λ_1. However, some of its roaming packets could also reach the borders of Λ_2 before consuming all of their relay credit. Thus, presence of the new sensor is advertised through the union of nodes of Λ_1, Λ_2, and in the *relay credit* > 0 zone. Based on the above setting, C_1 and C_2 start calculating (1), or (2), or both. Note that, C_3 might also receive the roaming packets of the new sensor if it is placed in $\Lambda_3 \bigcap \Lambda_1$ or $\Lambda_3 \bigcap \Lambda_2$, but it will not calculate (1) or (2) because Λ_3 is not listed among clusters in the packets.

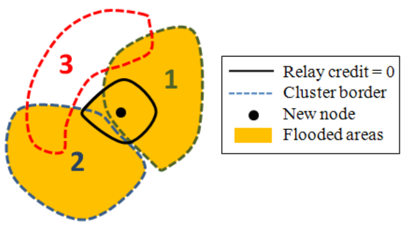

Figure 7. Effect of *relay credit* when a new node is joined

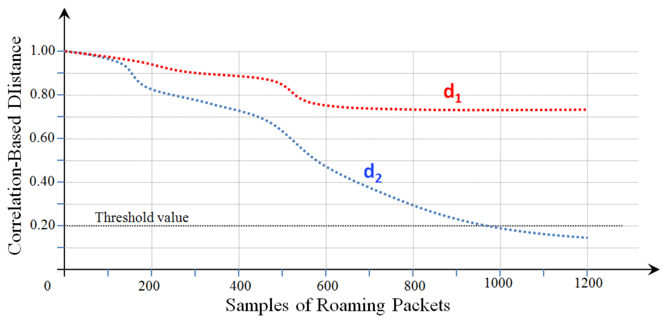

Figure 8. Development of usefulness measure (1) in C_1 and C_2 during identification process

Fig. 8 shows how the correlation-based distance measures (1) or (2) might develop over time in C_1 and C_2. It is assumed that the new sensor generates 1200 roaming packets.

After collecting sufficient samples at the cluster heads, each C_i decides whether the new sensor should become a member of their cluster or not. In Fig. 8, C_2 concludes that the new sensor is relevant well before the roaming packets are discontinued, but C_1 finds the new sensor irrelevant to the control performance of its internal model. C_2 continues the joining process by sending a *join* request to the new sensor. The data packet which contains the join request is in the following form, shown in Fig. 9.

010	1	Sensor Id.	relay credit	cl. Id. 1

Figure 9. Join request from cluster head to a sensor

Note that, this packet is in ”steady operating mode”, which means no relay credit is deducted unless it leaves the source cluster, Λ_2 in our example. This mechanism is useful when the new node is only reachable via nodes of other clusters.

After sending all roaming packets, equal to 1200 in our example, the new sensor applies received join requests. Join requests arrive at the sensor node asynchronously. Therefore, the node should continually accept join requests, at least until a pre-defined time. If no join request is received, the sensor starts another round of generating roaming packets.

If a group of sensors are deployed simultaneously, the ones which have other cluster members in their vicinity will find their clusters earlier. The others that do not have any neighbor operating in steady mode, may not inherit cluster tags, hence their flooding domain is limited to the *relay credit* > 0 zone.

The above procedure is slightly different for a new actuator. Each newly turned on actuator applies a pre-specified control sequence for the purpose of sufficiently exciting the plant and creating measureable outcomes. Simultaneously, it publishes roaming packets as shown in Fig. 10, which contain current value of the actuator output.

111	0	Actuator Id.	relay credit	no. of remained roaming packets	DATA

Figure 10. Packets generated by an actuator when it is just turned on

The cluster heads which receive these packets, use the DATA field in evaluating (5) similar to what was shown in Fig. 8. When the roaming packets are discontinued – meaning that the actuator is waiting for the decision – each cluster head returns the calculated *usefulness measure* to the actuator by packets shown in Fig. 11.

100	1	Actuator Id.	relay credit	cl. Id. 1	DATA

Figure 11. Packets that return usefulness of utilizing an actuator in a cluster

The actuator waits for a certain time to receive evaluation results from all involved cluster heads. Then it compares the received usefulness values, which are embedded in DATA fields, and selects the cluster that has returned the highest value. If at least one evaluation result is received, the actuator chooses its own cluster and sends a join request to that cluster as illustrated in Fig. 12. If no evaluated usefulness measure is received, the above procedure starts from the beginning.

110	1	Actuator Id.	relay credit	cl. Id. 1

Figure 12. Join request from an actuator to a cluster head

Note that, when a new sensor is added, it receives individual join requests from controllers. But when a new actuator is added, it sends the join request to a single controller.

IV. Performance Related Remarks

Remark 1: Unlike other clustered flooding-based routing protocols whose acceptable performance depend heavily on the optimal choice of the number of clusters and the thoughtful selection of cluster heads [2], [3], these parameters are pre-defined in our protocol because all of the controller units ($C_j, j = 1...n$) and only the controller units are cluster heads. Moreover, the controller nodes are fixed in the whole lifetime of the network.

Remark 2: Another issue is the influence of lower layer protocols on performance of the routing protocol. Unlike [2], that has utilized a TDMA-based MAC in sake of energy-efficient collision-free transmissions, the MAC layer in our system cannot accommodate a deterministic reservation-based protocol. It is mainly due to the constrained coverage range of nodes which does not guarantee existence of direct links between cluster heads and every member of the cluster to schedule a frame-based MAC. After all, it is a prerequisite for framed MACs that all of nodes can be accessed from a single base station for scheduling purposes. Otherwise, many time frames must be kept unused and reserved for future extension, as in Time Synchronized Mesh Protocol (TSMP) [8].

Our routing solution may be built either on a contention-based or a preamble-sampling MAC which are inferior to deterministic MACs in terms of energy efficiency and end-to-end latency if data transmission among nodes is frequent.

Therefore, the main benefit of our routing solution is to pick the members of each cluster so prudently such that it results in the minimum number of nodes in a cluster, and more efficient flooding in clusters.

V. Conclusion

In this paper, we have proposed a method to form clusters of nodes to be utilized by a flooding-based routing algorithm in decentralized wireless networked control systems. Our cluster formation method is data-centric and originates from the requirements of the control application. It makes use of model-based correlation estimation between new nodes and the existing model of the system. Furthermore, we have proposed to exploit non-member nodes in providing connectivity among nodes of an individual cluster. To this end, we have suggested to use an arbitrary routing metric, e.g. hop count. Operation of the proposed clustering mechanism is described in details.

References

[1] T. Watteyne, A. Molinaro, M. G. Richichi, and M. Dohler, "From MANET to IETF ROLL standardization: A paradigm shift in WSN routing protocols," to appear in IEEE Communications Surveys & Tutorials.

[2] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, October 2002.

[3] O. Younis and S. Fahmy, "Heed: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366–379, October-December 2004.

[4] IETF ROLL: routing metrics used for path calculation in low power and lossy networks. [Online]. Available: http://tools.ietf.org/wg/roll/draft-ietf-roll-routing-metrics/

[5] J. Kulik, W. Heinzelman, and H. Balakrishnan, "Negotiation-based protocols for disseminating information in wireless sensor networks," *Wireless Networks*, vol. 8, no. 2, pp. 169–185, March-May 2002.

[6] T. Knudsen, J. Bendtsen, and K. Trangbaek, "Awareness and its use in incremental data driven modelling for Plug and Play Process Control," to appear in European Journal of Control, 2011.

[7] IETF ROLL: IPv6 Routing Protocol, draft standard. [Online]. Available: http://tools.ietf.org/wg/roll/draft-ietf-roll-rpl/

[8] K. S. J. Pister and L. Doherty, "TSMP: time synchronized mesh protocol," in *Proceedings of the IASTED International Symposium on Distributed Sensor Networks*, November 2008.