Increasing Communication Reliability in Manufacturing Environments

Milena Radenkovic
School of Computer Science
The University of Nottingham
Nottingham, United Kingdom
mvr@cs.nott.ac.uk

Ivaylo Kostadinov
IEEE Member
ivaylo@kostadinov.name

Bartosz Wietrzyk
IEEE Member
bartosz@Wietrzyk.name

***Abstract*—This paper is concerned with low cost mechanisms that can increase reliability of machine to machine and machine to cloud communications in increasingly complex manufacturing environments that are prone to disconnections and faults. We propose a novel distributed and cooperative sensing framework that supports localized real time predictive analytics of connectivity patterns and detection of a range of faults together with issuing of notifications and responding on demand queries. We show that our Fault and Disconnection Aware Smart Sensing (FDASS) framework achieves significantly lower packet loss rates and communication delays in the face of unreliable nodes and networks when compared to the state of the art and benchmark approaches.**

Keywords — Opportunistic Networks, Ad hoc networks, Self-organized networks.

I. INTRODUCTION

When breakdown occurs (or is about to occur) in a complex manufacturing plant, rapid fault identification (or prediction), repair and recovery is crucial for avoiding large losses. Today's manufacturing environments typically rely on either sensing aimed at monitoring of simple manufacturing processes with infrequently changing conditions or on non-real-time cloud data mining for modelling and prediction of various events [1][2]. Many application-specific monitoring techniques and prototype systems have been proposed and developed such as multi-sensor approach to drill wear monitoring [3], a neural network-based multi-sensor system for monitoring the conditions of cutting tools [4][5], multi-sensor based monitoring of gear tooth fatigue for predictive diagnostics [6] and neural networks with multiple feature sets extracted from sensor networks for tool wear monitoring of turning [7]. Newly emerging work in [1] describes current best practices that extend simple on-line diagnostics provided by the equipment manufacturers for rapid diagnosis after failure. Reference [1] proposes an analytical approach for similarity-based prediction of manufacturing system performance where the authors compare similarity of the most recent performance signatures with the known degradation patterns available in the historical records. However these techniques have inherent limitations as they do not consider that the underlying end to end network connectivity may vary dramatically. Such solutions may miss on events and data that do not get delivered to the cloud due to network disconnections and thus result in incomplete diagnosis [8].

We argue that enabling continuous resource and data monitoring with limited maintenance and low deployment cost should constitute an integral part of any reliability mechanism. This paper describes the design and implementation of fault and disconnection aware smart sensing (FDASS) framework that enables continuous monitoring of data and resources. FDASS enables 1) fully distributed predictive analytics for detection and identification of a range of different faults and network disconnections that are implicit in highly automated plants, 2) multipath adaptive routing between sensors, machines and the cloud, and 3) storing of rich historical data in the cloud, issuing notifications and supporting on demand queries. We perform extensive experiments with 1250 raw heterogeneous sensors, 9 gateways that can aggregate and process sensor data, a cloud backend and three different kinds of mobile nodes that can store and query data on demand. We simulate a arrange of different faults divided in three categories to show that FDASS is successful at minimising negative impact of faults and disconnections in the plant and that it outperforms competing protocols across multiple metrics. In our previous work [9] we have shown the usefulness and feasibility of real time analytics in large complex systems (e.g. 540 nodes over multiple months). However, in this paper, the scale of the network and frequency of updates are considerably higher, resources lower, and it is a highly challenging problem to investigate how self-organization can improve the reliability of such fault and disconnection prone complex dynamic sensors networks that are integral part of these systems. After a brief overview of related work in Section II we provide detailed architectural and functional description of FDASS in Section III. Section IV explains our evaluation methodology and discusses results across a range of metrics before outlining future work in the conclusions in Section V.

II. RELATED WORK

This section briefly reviews most recent advances in data forwarding (with and without replication) in disconnection prone networks. Recent work in disconnection tolerant networks was concerned with buffer management [10][11][12][13], replication management [9][13] and distribution [9][11][14]. Reference [14] is a benchmark quota based routing protocol with replication that aims to forward more copies of a message to nodes that are core to the network but it does not check how fault prone these nodes are. Reference [9] proposed and examined several combined

connectivity and resource heuristics in order to detect congested parts of the network and move the traffic away towards less congested parts. While it increased packet delivery and lowered delays, it did not consider the possibility of faulty nodes. None of this work has considered the environment that we consider here which has high data rates, requires low delay and requires high precision.

[15] propose mechanism of resource pooling by harnessing multipath-capable end systems. In our work, the nodes benefit from pooling the store and forward capacity in in the network. More specifically, if traffic is spread across the resources of some faulty nodes and their neighbours, then traffic should identify quickly the affected regions, move away, provide notifications and utilise available and reliable parts of the network. Our FDASS combines heuristic function is at the core of our adaptive forwarding protocol that is dynamic and flexible as it operates as fault prediction and detection protocol we well as congestion aware protocol.

There is a body of work on Passive Clustering (PC) in ad hoc network relying on formation of clusters (e.g. [1][16]). Approaches that require proactive exchange of control messages to maintain the virtual infrastructure are less optimal for our scenario than passive clustering techniques (PC) due to limited resources. In PC the first node to broadcast its state is assumed to be a cluster head. All other nodes within its range become either gateways (that links multiple cluster together) or ordinary nodes. The node can become a gateway when it has sufficient resources. In our paper we use a variant of this technique when the contention among sensors is very high after a disconnection of fault has been discovered.

III. Fault and Disconnection Aware Smart Sensing

We propose an intelligent framework that can improve reliability of the manufacturing plant in the face of varying network connectivity and non-uniform distribution of different types of faults in the network. Fault and Disconnection Aware Smart Sensing framework (FDASS) is able to detect and identify misperforming nodes in a fully distributed fashion in order to isolate them, reroute the traffic away from them and notify the sinks about the type, location and time of the failures. FDASS builds on and extends multi-path transport approaches such as [9][13][17][18][19] to combine fault analytics layer and resource analytics layer with the complex heterogeneous network topology in dynamic manufacturing environments as shown in Fig. 1.

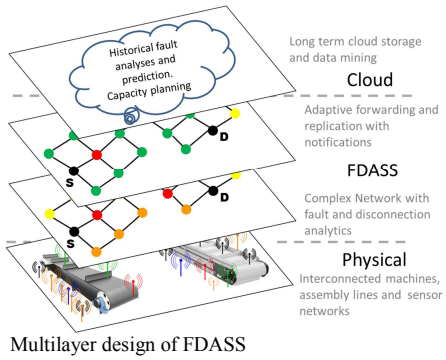

Fig. 1. Multilayer design of FDASS

FDASS works as a local adaptive fault-aware forwarding and replication protocol that diverts the load from its static path at times of faults and congestion, and directs it via a different path that avoids the faulty nodes and regions. In Section IV we show it decreases the load on highly central nodes while managing to have high success ratios and not introducing significant end-to-end delays. FDASS adaptively decides on the optimal number of replicas to forward in order to increase reliability of the system. We briefly describe the analytics that FDASS performs across multiple metrics below.

Retentiveness [9] refers to the node's available storage for the new packets that are sent to them. Retentiveness (*Ret(X)*) is important because of potentially very limited node buffer sizes in complex sensor networks and the possibility of faults. Retentiveness is calculated as an exponentially weighted moving average (*EWMA*) of a node's remaining buffer storage as given in Formula 1.

$$Ret(X) = B_c(X) - \sum_{i=1}^{N} m^i_{recieved}(X) \quad (1)$$

Where $B_c(X)$ is the size of the old buffer and $m^i_{recieved}(X)$is the size of the received message.

Receptiveness (Rec) [9] is complementary to Retentiveness and refers to the node's ability to receive packets and forward them on. This is important as increasing in-network delays is an indication of a possible fault or congestion. Formula 2 shows Receptiveness is the total current message delay, calculated as the sum of differences between the current time (T_{now}) and the time each message was received ($M^i_{recieved}$).

$$Rec(X) = \sum_{i=1}^{N}(T_{now} - M^i_{recieved}(X)) \quad (2)$$

Fault Rate and Congesting Rate (FR and CR) refer to measures of how fast a node is likely to become faulty and congest respectively. These are core to this proposal and help ensure its reliability and stability. Fault Rate signal as given in Formula 3 indicates the likelihood of the nodes getting faulty and causing inaccurate information reporting and disconnections. Each node that has sufficient resources keeps track of the percentage of time it has been faulty ($T_{\%f}(X)$) in Formula 3a, and the average time between its faults reoccurring ($T_{Fe}(X)$) in Formula 3b.

$$FR(X) = \frac{T_{\%f}(X)}{T_{fe}(X)} \quad (3)$$

$$T_{\%f}(X) = 100 \cdot \frac{T_{FaultyNode}(X)}{T_{TotalTime}(X)} \quad (3a)$$

$$T_{fe}(X) = \frac{1}{N} \cdot \sum_{i=1}^{N} T_i end(X) - T_i start(X) \quad (3b)$$

We described congestion rate (CR(X)) in [9] and showed how it indicates the likelihood of traffic spikes that can cause the message to be dropped in. In FDASS, each node keeps track of it its congestion rate to avoid congesting individual nodes and regions of the network.

We define Region (*Reg*) as a network consisting of a single node together with the nodes that are its n-hop dynamically changing neighbours. FDASS dynamically

aggregates Retentiveness, Receptiveness, Fault Rate and Congesting Rate for each Region to form dynamical regional perspective of the network. FDASS uses EWMA to aggregate fault and resource analytics information for each node (α is typically 0.05) as shown in Formula 4.

$$WReg_f\big(X, N_i(X)\big) = (1-\alpha)\cdot WReg_f(X) + \alpha \cdot f(N_i(X)) \quad (4)$$

The aggregate FDASS heuristics that integrates both node and regional resource analytics is given in Formula 5.

$$FDASSHeur_D(X) = \sum_{h\in H} Heur_h(X) \quad (5)$$

Note that we consider congestion imposed by storage/buffer limitations while network contention is out of the scope of this paper. Control information that nodes need to exchange is kept low as it includes only scalar values that represent end statistical analyses performed by the nodes.

Without loss of generality, we consider three types of faults common to manufacturing environments that FDASS is able to detect, isolate and send notifications about: ABFR refers to nodes reporting with abnormal frequencies (e.g. nodes reporting slower or faster than normal). This can be caused by calibration failures, poor lighting, part reflectivity, and unknown failure. The receiving node keeps track of timestamps of readings together with the normal frequency range set for that sensor. Through comparisons with expected/normal frequency and calculated/observed frequency the receiving node is able to indentify ABFR faults. ABSV refers to nodes reporting abnormal sensor values (e.g. sensors values outside one standard deviation of the average readings). This can be caused by sensor failure, inadequate sensing capability, ineffective maintenance, noise. The receiving node keeps track of received sensor readings together with the normal sensor readings range for that sensor. Through comparison with expected/normal reading and received readings the receiving node is able to discover ABSV faults. NRES refers to nodes being unreachable and non-responsive. This can be caused by collisions, wear, contamination, moisture ingress.

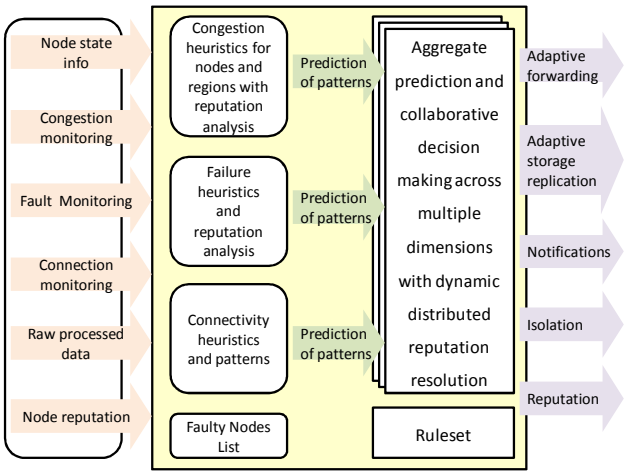

Fig. 2. FDASS Architectural Overview

FDASS stores fault profiles that are tuples of four fault attributes: (fault-category, fault-location, fault-time-interval, fault-summary-vector). Nodes exchange their knowledge about the detected faults with other nodes so that they can learn faster about multiple different faults occurring across the network so that they can faster decide on the appropriate action to be taken in order to minimise the negative impact of the faults. Different types of faults can have different degrees of severity that may require different responses. FDASS enables distributed continuous monitoring of the network behaviour and reported sensor data in order to provide fast detection of faults and perform suitable actions. For example, if the neighbouring nodes calculate deviation from the expected value (ABSV Fault) that one node is reporting, A FDASS node will promptly change the route to avoid that node, discard its data and notify the local network The architectural overview of FDASS is given in Fig. 2 and its pseudo code is in Fig. 3.

Pseudo code FDASS

```
List Faulty = {}
// identify and avoid faulty nodes
For each Neighbour do:
    If Faulty.Contains(Neighnour) Then:
        Skip
    End If
    // monitor and learn about network
    Monitor(Neighbour.Faults)
    ExchangeRepuation(Neighbor)
    ExchangeFaultInfo(Neighbor)
    // fully distributed reputation management
    If ( Neighbour.isFaulty) Then:
          Reputation.Resolve(Neighbour) {
          Faulty.add(Neighbour)
          Exchange,ReputationResolved(Neighbour)
          Exchange.FaultData(Neighbour)
          Skip
    End If
    // gather total stats and update heuristics
    Monitor(Neighbour.Congestions)
    Monitor(Neighbour.Connectivity)
    CalcCongHeur (Neighbour.Cong)
    CalcConnHeur(Neighbour.Conn)
    FDASSHeur=exchFDASSHeurInfo(Neighbour)
    FDASS.Insert(Neighbour)
End For
// adapt routing and replication behaviour
For Each node in FDASS do:
    Neighbour.local.FDASSregion.form()
    FDASSreg.ForwRate =CalcFrwRate(FDASScluster)
    FDASSreg.ReplRate = CalcRplRate(Neighbour)
    FDASSreg.Sent = TransmitData(FDASSreg,
       FDASSreg.FrwRate, FDASScluster.RplRate)
End For
```

Fig. 3. FDASS pseudocode

FDASS uses Eigen Vector and EigenTrust [20] to represent nodes' agreements or disagreements over potentially faulty nodes. The trust value of a potentially faulty node is resolved as the weighted sum of multiple nodes' observations of the node divided by the total number of nodes as:

$$Tm = \frac{\sum_{i=1}^{|V|} w_i \cdot T_i}{|V|} \quad |v \geq 3 \qquad (6)$$

where T_i is observation of the node i for the trust value of a faulty node m, and w_i is the weight of this observing node i. The minimum required number of sampling nodes is 3, has been experimentally determined to achieve best trade-off between accuracy of fault awareness and delays imposed. FDASS identifies and uses nodes that have both high centrality and reputation to both increase reliability and accelerate reputation convergence. However, we assume that even the highly important and central nodes can get faulty and we aim to minimise the potentially detrimental effects that this can have on the manufacturing plant (e.g. in terms minimising data loss rates).

With FDASS we show that the failure of such significant nodes gets discovered very quickly, so the losses get minimized and the faulty nodes get isolated. Each node holds N neighbour trust records where N is limited by the node's memory and computational power. Nodes with higher trust values and centrality hold higher number of trust records about other nodes. Nodes use expiration timer to achieve trade-offs between responsiveness, stability, and resource efficiency. FDASS performs deviation test to ensure the consistency between different nodes' point of view on the same node as described in [21],[22]. In this way, FDASS represents significant extensions to previous work such as [9] and proposes completely new fault profiles and reputation management, new implicit heuristics and new accurate analytics based on statistics and measurements.

IV. Evaluation Methodology and Results

A. Example Scnario

Consider a complex heterogeneous cyber physical network architecture that drives a range of processes of a large factory plant. We assume that there are three types of mobile units on the shop floor with different capabilities and different range of sensors: automated guided vehicles (AGV) that can act as data-mules between different sections of the manufacturing plant, Floor Engineers (FE) and Assembly Lines (AL).

AGV is a sink with moderately large storage (50MB) and high processing power that can forward the data to the cloud or to the Sensor gateways by exploiting its physical mobility as well as issues queries about state of different regions of the plant that is within its range. Sensor gateways have storage between 250 and 500MB. Floor engineer (FE) is assumed to carry an electronic inspection unit (a tablet or, a smartphone) that can send in situ queries to nearby nodes to enquire about their status and also store and process limited amount of data. We consider five cells with 250 heterogeneous sensors and 5 assembly lines each. The sensors generate a reading every second with an average size varying between 1KB and 1MB. Each assembly line contains static sensors attached to the line that can gather data from one end of the assembly line and carry them to the other end of the assembly line. Each sensor cell can be connected via a RF, 802.11 Ethernet, Zigbee, or 802.15.4/6LoWPAN link to its default processing and aggregating gateways and can exchange on-demand data with them. All gateways can communicate with each other and the Clouds. They utilize wireless (or wired) links. We assume typical speeds of 15-20m/s for Assembly line, 2m/s-5m/s for the Floor Engineer and 7-10m/s for AGV. The communication range of sensors is assumed to be 50m for RF and 100m for wireless Ethernet.

B. Experiment set up and comparison with benchmark and state of the art protocols

We analyse the influence of changing numbers and distribution of faulty nodes in the face of increasing percentage of faulty nodes with the following steps of increase: 10%, 20%, 30%, 40%, 50% and 60%. More specifically, we incrementally expand the cells that are affected by faults by increasing the probability of fault occurring in each cell. For example in the early stage of the experiment, faults occur in a single cell only, then the faults expand to a neighbouring cell, and this process continues until the faults may occur throughout all cells. We argue that this choice of scenarios allows us to test our protocol in a range of different scenarios applicable to the real world scenarios such as faults that can propagate via Assembly Line or processes in which one faulty cell controls another cell. We show that self-organised sensing and self-cooperation in FDASS is essential for high levels of reliability and scalability. Assumption of non-adaptive topologies can be damaging as they results in high levels of faulty nodes not being detected in time or not at all. We carry out comparative performance evaluation of FDASS, state of the art cloud that performs similarity based time series prediction of faults as in [1], and benchmark hierarchical approaches that report to the cloud but do not perform automated fault detection and analysis. All of our experiments are conducted in the ONE simulator[23].

C. Results

1) Failure Detection

Fig. 4 shows that local failure detection rate in FDASS is always 100%. With the cloud approach some packets get dropped due to congestion or disconnection when the fault is being reported, thus some faults do not get identified accurately; with increased congestion levels the detection rate goes down to 42%.

Traditional hierarchical approaches do not detect faults as they do not incorporate fault analysis and prediction. This shows that FDASS local processing and management is more effective than offloading and subsequent processing in a centralised manner.

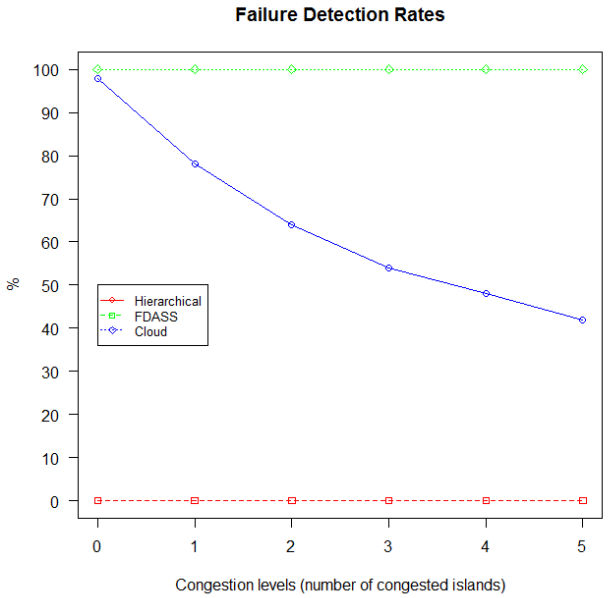

Fig. 4. Failure Detection Rates

Fig. 5 shows that delays for local fault discovery in FDASS of all three types of faults are very low (under 1 second). This means that there is a very small additional delay imposed by nodes coordination of their views on the faults even in the face of high congestion. Reporting the faults over unstable paths for the cloud approach takes increasingly long time (ranging from above 2 sec to 14 sec) as the congestion increases. Benchmark hierarchical approaches impose no reporting fault delays as they do not identify any faults.

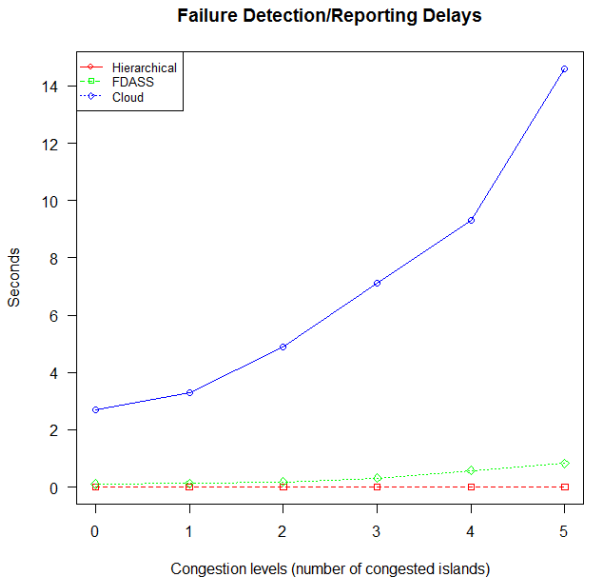

Fig. 5. Failure Detection Reporting Delays

2) Failure Analysis

We show analyses of three different types of failure detection for different congestion levels in the face of non-uniform failure distribution across the whole network. Fig. 6 shows the number of different failure types detected. We observe that the likelihood of faults and congestion are positively correlated i.e. increasing node failures are increasing as congestion is increasing and vice versa.

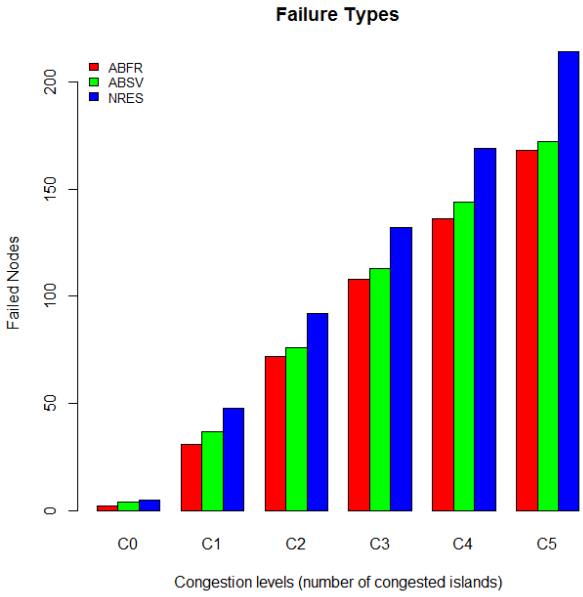

Fig. 6. Failure types detection

When a failure is detected, FDASS reports on the type of node on which the failure occurred and the type of the node that detected the failure first. Table 1 shows that the majority of faults happen with the sensor nodes, followed by aggregating gateways. Processing gateways are least llikely to fail. We show that even though all nodes have the ability to detect some failure (including sensors that can detect that they cannot contact the Aggregating gateways), Aggregating gateways detect the majority of failures as they can detect both sensor failures and processing gateways failures. Processing gateways can detect failures of Aggregating gateways, e.g. when they have not received data for extended periods of time but sensors identify them first as they detect they cannot upload data. Only in cases when Aggregating gateways begin reporting with changed frequency or unexpected data to the Processing gateways, then the Processing gateways will be first to identify these types of faults. From our experiments, Table 1 shows that ABFR ranges from 2 through 108 to 168 for low, medium and high congestion levels respectively. ABSV ranges from 4 through 113 to 173 for low, medium and high congestion levels respectively. NRES ranges from 4 through 132 to 214 for low, medium and high congestion levels respectively.

TABLE I. FAILURE OF NODES BY TYPES

Congestion	Sensor Fault Location/Fault Detector		Aggregator Fault Location/Fault Detector	
0	0	0	0	0
1	128	0	0	128
2	252	1	1	252
3	371	1	1	372
4	492	2	2	493
5	616	3	3	618

Fig. 7 shows the failure impact different types of nodes have on the network performance. The impact is measured in terms of the number of lost messages. We observe that individual sensor failures have very little impact while the effects of aaggregating and processing gateways is much more significant. In all cases FDASS outperforms Hierarchical benchmark approach. We observe that FDASS loses on average 10 times less packets compared to the hierarchical approach. For example FDASS aggregating gateway loses from 3428 to 240949 packets for the first and forth congestion levels respectively while Hierarchical Aggregating gateway loses from 150000 to 600000 for low and medium congestion levels respectively.

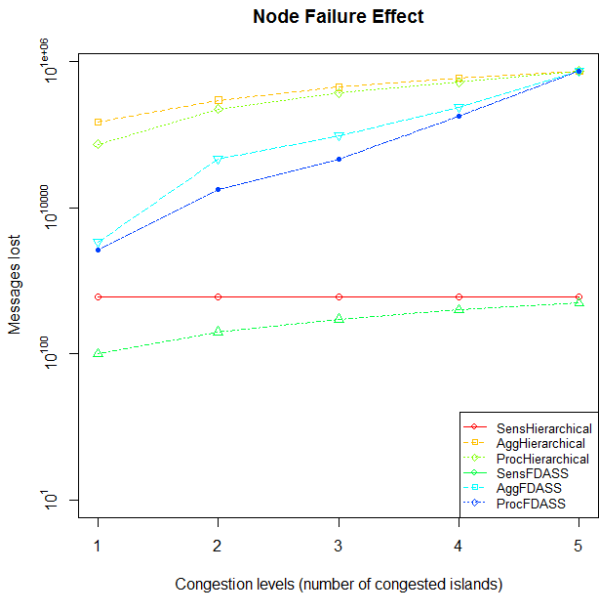

Fig. 7. Node failure effect

3) Performance Statistics

Fig. 8 shows the comparative performance of FDASS and hierarchical protocols when supporting interacting communication with the mobile nodes Assembly Line, Floor Engineer and AGV, e.g. uploading on demand data and answering queries.

The highest success ratio is shown for the interactive communication with AssemblyLine over FDASS (ranging between 95% and 58%).

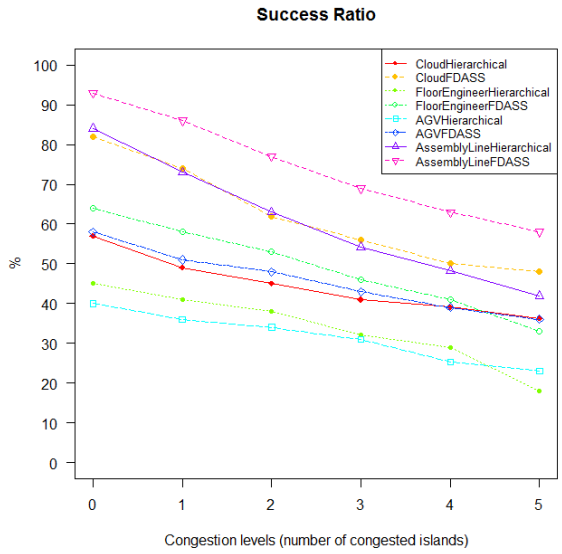

Fig. 8. Success rates with congestions levels increasing

This is followed by the interactive communication with the Cloud over FDASS communication (ranging between 82% and 48%). Hierarchical (non fault aware and non congestion aware) communications with AssemblyLine and the Cloud perform more than two times worse than FDASS. AssemblyLine over FDASS has the best performance because it is very well connected with all sensor nodes and can receive packets directly from them even in cases of Aggregating or Processing gateways faults or congestions. Interactive communication with AGV over FDASS has the lowest performance of all intelligent protocols and even lower that some of the hierarchical protocols; this is due to the short periods of connectivity with the AGV separated by long periods of isolation as the AGV moves across the plant.

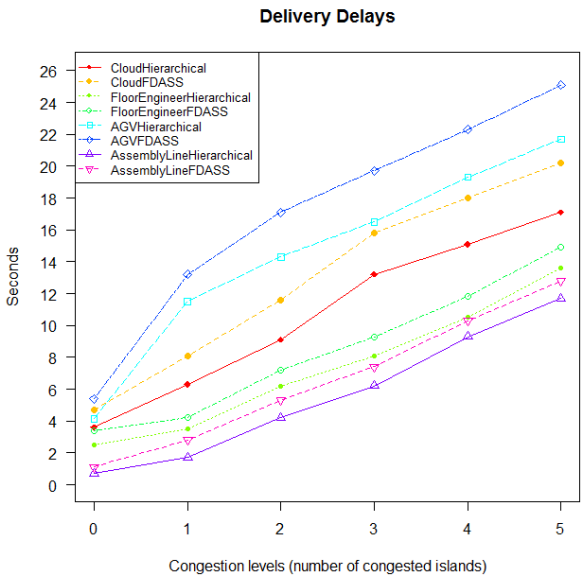

Fig. 9. Delays with congestion levels increasing

Fig. 9 shows that interactive FDASS communication with the three sinks have slightly higher delays (around 5% on average) compared to the corresponding hierarchical protocols; this is due to the extra time taken to reroute the packets away from faulty or congested nodes. These delays are however negligible compared to the added benefit of increased success ratio of interactive on demand communication.

V. Conclusions

In this work we proposed FDASS framework that manages to detect and avoid faulty nodes in the disconnection prone heterogeneous sensor networks. Our approach was to unify adaptive forwarding and replication into a common fault aware and congestion aware framework that decreases the load on the network and offloads the traffic to the parts of the network that are more reliable and less congested. We achieve this by using a local based implicit heuristic based on fault and resource analytics that extends the previous work on static hierarchical architectures. Node fault rate and profiles heuristics allow identification and isolation of faulty nodes while node resource driven heuristics allows FDASS to adapt to heterogeneous nature of the sensor network. Network driven heuristics allows tolerance to disconnections and adaptation to congesting rates.

FDASS significantly outperforms cloud and hierarchical communication approaches across several criteria. FDASS has significantly higher success ratios of detecting different kinds of faults compared to the cloud approach that missed almost 50% of the faults at times of high congestion in the network and in the face of distributed faulty nodes. This is due to the cloud not receiving full and timely information from the parts of the network that are unreliable.

As part of our future work, we plan to test FDASS in a heterogeneous sensor network testbed comprising of multiple single-board computers (Raspeberry PI) and various sensors (temperature, magnetic, acceleration, cameras etc.) in a setup that closely resembles real world application scenarios. We believe that FDASS framework should constitute an important integral part for future smart manufacturing systems that move from M2M communications towards complex cyber-physical systems in addition to the one already proposed in [12],[24].

Acknowledgements

This work was supported in part by the FP7 "SelSus: Health Monitoring and Life-Long Capability Management for SELf-SUStaining Manufacturing Systems" project, 609382

References

[1] Alexander Bleakie, Dragan Djurdjanovic, Analytical approach to similarity-based prediction of manufacturing system performance, Computers in Industry, Volume 64, Issue 6, Aug 2013, Pages 625-633

[2] M. Hermann, G. (1990). Artificial Intelligence in Monitoring and the Mechanics of Machining. Computers in Industry vol. 14(1-3), 131-135

[3] NooriKhajavi, A., Komanduri, V. (1993). On Multisensor Approach to Drill Wear Monitoring. CIRP Annals vol. 42(1), 71 -74.

[4] Littlefair, G., Javed, M.A., Smith. G.T. (1995). Fusion of Integrated Multisensor Data for Tool Wear Monitoring.Proc. IEEE International Conference on Neural Networks 2, 734-737.

[5] Javed, M.A., Hope, A.D. (1996). On Line Multisensor System Monitors Machine Tool Wear. Noise and Vibration Worldwide vol. 27(5), 17-18.

[6] Gordon, G.A., Moose, C.A. (1998). Multisensor Monitoring of Gear Tooth Fatigue for Predictive Diagnostics.TriboTest vol. 4(4), 393-406

[7] Silva, R.G., Reuben, R.L., Baker, K.J., Wilcox, S.J. (1998). Tool Wear Monitoring of Turning Operations by Neural Network and Expert System Classification of a Feature Set Generated from Multiple Sensors. Mechanical Systems & Signal Processing vol. 12(2), 319-332.

[8] Hu W, Starr A, Leung A. A multisensor-based system for manufacturing process monitoring. Procs of the Institution of Mechanical Engineers Part B: Journal of Engineering Manufacture. 2001;215(9):1165-1175.

[9] Milena Radenkovic, Andrew Grundy, Efficient and adaptive congestion control for heterogeneous delay-tolerant networks, Ad Hoc Networks, Volume 10, Issue 7, September 2012, Pages 1322-1345

[10] S. Burleigh, E. Jennings, and J. Schoolcraft. Autonomous congestion control in delay-tolerant networks. American Institute of Aeronautics and Astronautics, 2007.

[11] J.M. Pujol, A.L. Toledo, and P. Rodriguez. Fair Routing in Delay Tolerant Networks. In Proceedings of IEEE INFOCOM, 2009.

[12] Aline Carneiro Viana, Marcelo Dias de Amorim, Coverage strategy for periodic readings in robotic-assisted monitoring systems, Ad Hoc Networks, Volume 11, Issue 7, September 2013, Pages 1907-1918

[13] M. Seligman, K. Fall, and P. Mundur. Storage routing for dtn congestion control. Wireless Communications and Mobile Computing, 7(10), 2007.

[14] S.C. Nelson, M. Bakht, R. Kravets, and A.F. Harris. Encounter: based routing in DTNs. ACM SIGMOBILE Mobile Computing and Communications Review, 13(1):56–59, 2009.

[15] D. Wischik, M. Handley, and M.B. Braun. The resource pooling principle. ACM SIGCOMM Computer Communication Review, 38(5):47–52, 2008.

[16] B. Wietrzyk, M. Radenkovic, and I. Kostadinov, "Practical MANETs for Pervasive Cattle Monitoring", ICN 2008, Cancun, Mexico, pp 14-23

[17] Haque, Md.Emdadul,Wei, Fan,Gouda, Takehiro, Lu, Xiaodong Mori, Kinji, "Autonomous Online Expansion Technology for Wireless Sensor Network Based Manufacturing System", Lecture Notes in Computer Science, Autonomic and Trusted Computing, 2011, Volume 690

[18] L. Q. Zhuang, D. H. Zhang, M. M. Wong, Wireless Sensor Networks for Networked Manufacturing Systems, Factory Automation, 2010-03-01

[19] N. Thompson, S. Nelson, M. Bakht, T. Abdelzaher, and R. Kravets. Retiring replicants: Congestion control for intermittently-connected networks. In Proceedings of Infocom, 2010.

[20] M.T. Schlosser, "The EigenTrust Algorithm for Reputation Management in P2P Networks," ReCALL, 2003.

[21] S. Buchegger and J.L. Boudec, "A robust reputation system for peer-to-peer and mobile ad-hoc networks ", proc. of P2PEcon, 2004.

[22] P. Michiardi and R. Molva, "CORE: A Collaborative Reputation Mechanism to enforce node cooperation in Mobile Ad hoc Networks", Proc. IFIP CMS, 2002.

[23] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. 2009. The ONE simulator for DTN protocol evaluation. In Proc (Simutools '09). ICST.

[24] Wan Jiafu, Chen Min, Xia Feng, Di Li, Zhou Keliang, From machine-to-machine communications towards cyber-physical systems, Computer Science and Information Systems 2013 Volume 10, Issue 3