This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2018.2859307, IEEE Transactions on Network Science and Engineering

Adaptive and Fault-tolerant Data Processing in Healthcare IoT Based on Fog Computing

Kun Wang, *Senior Member, IEEE,* Yun Shao, Lei Xie, *Member, IEEE,* Jie Wu, *Fellow, IEEE,* and Song Guo, *Senior Member, IEEE*

Abstract—In recent years, healthcare IoT have been helpful in mitigating pressures of hospital and medical resources caused by aging population to a large extent. As a safety-critical system, the rapid response from the health care system is extremely important. To fulfill the low latency requirement, fog computing is a competitive solution by deploying healthcare IoT devices on the edge of clouds. However, these fog devices generate huge amount of sensor data. Designing a specific framework for fog devices to ensure reliable data transmission and rapid data processing becomes a topic of utmost significance. In this paper, a Reduced Variable Neighborhood Search (RVNS)-based sEnsor Data Processing Framework (REDPF) is proposed to enhance reliability of data transmission and processing speed. Functionalities of REDPF include fault-tolerant data transmission, self-adaptive filtering and data-load-reduction processing. Specifically, a reliable transmission mechanism, managed by a self-adaptive filter, will recollect lost or inaccurate data automatically. Then, a new scheme is designed to evaluate the health status of the elderly people. Through extensive simulations, we show that our proposed scheme improves network reliability, and provides a faster processing speed.

Index Terms—Healthcare IoT, Fog Computing, Reduced Variable Neighborhood Search (RVNS), Data Load Reduction, Transmission Reliability

1 Introduction

In recent years, elderly people have witnessed improvements in healthcare IoT systems. These intelligent systems take a comprehensive remote health care by monitoring food quality, diet, daily exercise, physiological status, etc. of elderly people. Along with the developments, it also generates more demanding in terms of storing and processing huge amount of data with low latency. As a time-sensitive service, traditional cloud computing can hardly meet the requirement because sensor data takes too much time before arriving at core storage and processing nodes. To solve the problem, the idea of fog computing is adopted in recent years. Instead of sending data to core nodes, fog devices receive the sensor data and provide processing results for eHealth clients, e.g., local hospitals and health care providers, which diminishes the latency significantly. Nonetheless, many factors may restrict the development and implementation of the fog computing supported healthcare IoT systems. For instance, when sensors are deployed into everyday objects [1], the network scale will expand significantly, which is likely to result in huge amount of data. On the other hand, it can hardly to deploy powerful processors in fog devices. How to process large amount of complex data fast and efficiently with limited computing capability, is one of the most important issues to be addressed.

To deal with the big data processing problem, a typical approach is to break data into small pieces. Guided by this, a novel algorithm, Variable Neighborhood Search (VNS) [2], was proposed. In VNS, a global solution space is divided into a neighborhood structure consisting of several small changeable neighborhoods through some predefined criteria. Then, VNS calls subroutines to find better local optimal solutions from the neighborhoods to approach the global optimal solution, and reconstructs the neighborhoods iteratively until the number of neighborhood reconstruction or the running time of CPU reaches the maximum. The major issue associated with this method is that calling subroutines can be extremely time consuming. To address this problem, P. Hansen *et al.* [2] proposed a simplified version: Reduced Variable Neighborhood Search (RVNS). In RVNS, complex subroutines are replaced by the selection of a random point in each neighborhood, which means randomly selecting a local optimal. Thereby, RVNS simplifies computation complexity. This characteristic is quite suitable for healthcare IoT in several aspects. For example, the simplicity of RVNS facilitates rapid processing. Also, the collected data is not as meaningful as the data that may carry information indicating danger. The randomness of RVNS provides a chance that processor skips less meaningful data and focuses on the data that might be more valuable.

In this paper, we leverage the RVNS algorithm to refine valuable information from raw sensing data at fog devices. Specifically, we rearrange data processing sequence inspired by the neighborhood structure in RVNS, which differs from the traditional processing where First-in First-out (FIFO) queues [3] are normally deployed. In the queue, data processing algorithms take data with receiving sequence, which implies that the waiting time of the latest data is high if the

K. Wang is with the School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing 210003, China (e-mail: k-wang@njupt.edu.cn).

Y. Shao is with the Department of Computer Science, University of Southern California, Los Angeles, CA 90089, USA (e-mail: yunshao@usc.edu).

L. Xie is with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China (e-mail: lxie@nju.edu.cn).

J. Wu is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA (e-mail: jiewu@temple.edu).

S. Guo is with Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China (e-mail: song.guo@polyu.edu.hk).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2018.2859307, IEEE Transactions on Network Science and Engineering

data volume is large. To solve this problem, we design a new RVNS queue for fog devices. we divide the collected sensor data into several levels according to its range first. Then, neighborhood structures are constructed according to their levels, and an RVNS queue is formed based on these neighborhood structures. In the end, data in the RVNS queue is processed in sequence through a customized risk assessor with specially designed parameters.

However, there are several challenges to introduce RVNS into the data processing phase of fog computing based healthcare IoT. As a safety-critical system, an healthcare IoT system has to quickly retrieve and process accurate sensor data, and provide proper decisions [4]. During this process, any faults or failures may lead to serious problems for the elderly people. Therefore, how can the system collect data reliably is of utmost significance. To ensure reliable transmission, integrating some fault tolerant algorithms is necessary. In addition, how to balance resource utilization inside fog computing platforms is another issue [5]. Ultimately, in order to process the data, we need to design an efficient scheme. In this paper, we introduce modifications and shed light on their implementation in the RVNS algorithm, to jointly consider both reliability and efficiency requirements.

We primarily concentrate on the reliability of data processing in healthcare IoT based on fog computing, and design an RVNS-based sEnsor Data Processing Framework (REDPF) to reduce data load and refine results for eHealth clients. The contributions of our paper are summarized as follows:

- We develop a fault-tolerant mechanism to increase the reliability of data transmission between fog devices. The mechanism combines the thought of Directed Diffusion and Limited Flooding to ensure the reliability.
- We propose a self-adaptation scheme to dynamically optimize system resources. It can adjust parameters of the fault-tolerant mechanism based on current condition of fog computing platform, which improves the reliability for healthcare IoT further.
- We design a data analyzing scheme with an RVNS queue to respond to different kinds of requests from ehealth clients efficiently. By adjusting original terminal condition of RVNS algorithm, the new scheme fits well with the need of fog computing supported healthcare IoT.

The rest of the paper is organized as follows. In section II, we introduce state of the art on data processing and reliability for healthcare IoT systems, and current trends in fog computing. In section III, the overall design of the framework is presented. In the following three sections, we demonstrate three main functionalities of REDPF: Fault-tolerant Data Transmission, Self-adaptive Filtering and Data-load-reduction Processing, respectively. Simulation results are presented in section VII. Section VIII concludes the paper.

2 Related Work

2.1 Data Processing in healthcare IoT

A good volume of literature exists on data computing in healthcare IoT systems. Winkey *et al.* [6] improved the original sensor interface, and added a classifier to reduce data dimension. Magherini *et al.* [7] proposed an automated recognizer that records daily activities, which ensures the accuracy for sensor data through propositional temporal logic and model checking. J. Wang *et al.* [8] proposed a sound analyzing architecture to distinguish acoustical signals among noise. Recent studies on real-time data processing architectures and platforms [9-11] focus on improving the accuracy of data analysis. It is also important to consider environmental factors, such as, temperature or air pressure in order to further improve the accuracy. In this regard, existing studies can be divided into two categories. One tries to minimize the influence of environmental factors with more precise analysis algorithms or tools [12-14], the other involves collecting comprehensive data [15, 16]. However, both of these methods increase the workload and pressure on healthcare IoT system.

2.2 Fault Tolerant Transmission

Systems become unreliable when more environmental factors are involved, because some failing nodes have a higher possibility to become trunk nodes. Classical sensor fault tolerant algorithms can be categorized as follows. Firstly, there are some flooding-based algorithms [17], which relay packets to surrounding nodes. Another category is based on gradient broadcasting. This kind of algorithm first sets a gradient field in the network, and packets are relayed according to the gradient filed. A typical algorithm in this category is Directed Diffusion [18]. The last category is based on clustering hierarchy. These algorithms use the condition of clustering heads as a decision maker in relay, and the typical algorithm is Low-Energy Adaptive Clustering Hierarchy [19]. Despite the comprehensive studies regarding sensor fault tolerance, whether they are suitable for safety-critical systems, such as ambient assisted living deserves further discussion.

2.3 Fog Computing

Before fog computing is proposed, cloud computing plays an important role in on-demand data processing. Cloud computing can provide various service including Software-as-a-Service, Platform-as-a-Service, Infrastructure-as-a-Servce, Sensing-as-a-Service, etc. Compared with cloud computing, fog computing is a relatively new concept. It processes data at the edge of cloud to support low latency and geo-distribution services [20]. Even recently many researchers pay attention to fog computing architectures. For example, Hu *et al.* [21] proposed a hierarchical multi-access edge computing framework for vehicular network integrated with a specially designed protocol and millimeter wave communications, which improves performance in lots of network condition.

Introducing fog computing into healthcare IoT is not a new idea. Cao *et al.* [22] adopted fog computing to detect falling. Aazam *et al.* [23] optimized emergency alert service through fog computing. However, existed state of arts paid limited attention towards transmission and computing ability of fog computing itself.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2018.2859307, IEEE Transactions on Network Science and Engineering

3 System Model

We aim to implement a new framework for eHealth clients taking advantages of fog computing to flexibly fulfill their requirements on data gathering and data processing. In order to achieve this goal, we need to first ensure reliable communications between portable intelligent sensors and fog devices. It can be evaluated through completeness and correctness of the received data. Usually, transmission between portable sensors and storage nodes of a fog computing supported healthcare IoT system is implemented by wifi of some other highly reliable technologies, but problems may happen between the storage nodes and processing nodes inside the system. Additionally, we need a rule to measure the system conditions, and dynamically allocate system resources. In terms of big data processing, the fog devices need to run various functions for different eHealth clients, and it is possible that lots of data is waiting in the processing queue because the huge amount of data is straining processors. Furthermore, the following metrics are also important for processing performance.

- *Accuracy of processing results:* In healthcare IoT systems, the processing results directly reflect safety of the elderly. Some potential risk may be ignored if e-Health clients receive inaccurate results. Accordingly, there is a high accuracy requirement in distinguishing different living situations.
- *Time-efficiency:* To ensure the accuracy, healthcare IoT systems tend to collect more data to indicate a comprehensive living status. Also, For accurate results, more powerful and complex data analyzing algorithms are adopted. When a complex algorithm is run on a set of more data, the processing time will clearly be longer. On the contrary, eHealth clients promptly need the results from the analysis.
- *Data-Load-Reduction:* Since the final results are transmitted to eHealth clients for direct use, the data volume should be reduced for reliable transmission.

Fig. 1 shows the structures of the RVNS-based sEnsor Data Processing Framework (REDPF). The functionalities of REDPF can be summarized into three parts: Fault-tolerant Data Transmission, Self-adaptive Filtering, and Data-load-reduction Processing. Fault-tolerant Data Transmission is designed for receiving relevant data from the portable intelligent devices. Also, it checks completeness of the received data, and retrieves the lost data through a fault-tolerant mechanism. Then, sensor data will enter a self-adaptive filter. The filter is designed to ensure the effectiveness of data. In addition, a self-adaptation module is used to adjust the parameters of this fault-tolerant mechanism. Finally, an RVNS-based computing and analyzing (RCA) scheme is designed for dealing with the RVNS queue. RCA will generate results according to relevant pre-defined rules from eHealth clients. These results will provide overall evaluations of elderly people's living status, which will be transmitted to eHealth clients as feedback.

4 Fault-tolerant Data Transmission

In REDPF, the Fault-tolerant Data Transmission ensures transmission reliability between the storage nodes and the processing nodes. Currently, related studies have focused on data transmission in sensing level, but transmission inside a fog computing platform has received much less attention. Accordingly, we introduce a new fault-tolerant mechanism.

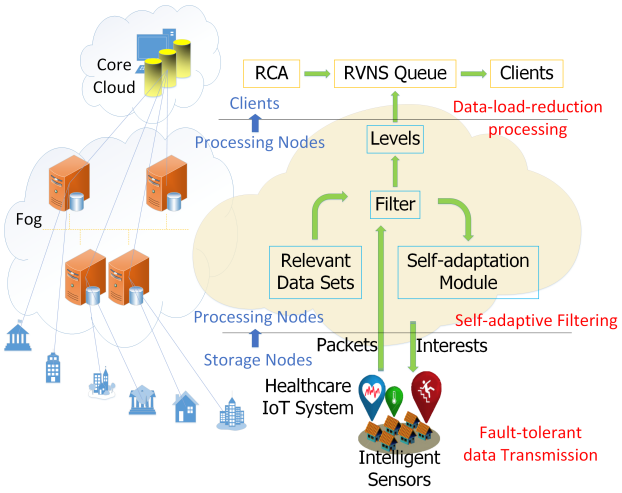


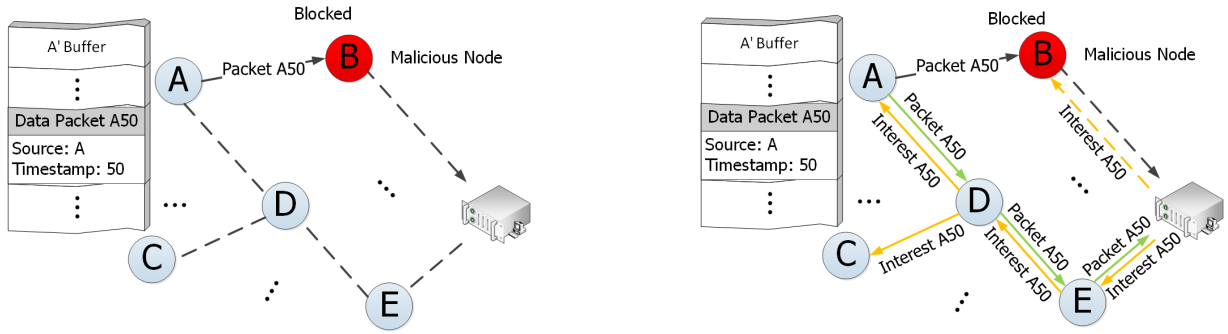
Fig. 1: Structure of REDPF

In a fog computing supported healthcare IoT system, fog devices consists of storage nodes and processing nodes, as shown in Fig. 2. A storage node is used to store sensor data collected from intelligent sensors. These fog storage nodes also upload their stored data to core cloud for future reference. When an eHealth clients propose a request, the processing nodes will retrieve relevant data from the storage nodes, and send back the refined results. As mentioned before, due to transmission between intelligent devices and fog devices are usually supported by some highly reliable technologies, we will assume sensor data can be transmitted to the storage nodes reliably.

However, with the expanding of deployment, a storage node may need several relay nodes to help relay towards the destination. When some relay nodes do not function, the connected links will also be broken, as marked in red in Fig. 2. We propose a fault-tolerant mechanism that takes the advantages of both directed diffusion and limited throughput, as shown in Fig. 3. Here, we assume all the storage nodes have more than one potential links to the processing node, and focus on the fault-tolerant mechanism in routing stage, rather than the self recover mechanism. Even though multiple links exist, there is no guarantee the mechanism can recollect all the lost packets if all the links are broken, as shown in the experiment.

For the fault-tolerant mechanism, we need a proper trace back strategy. At the beginning, to identify the storage nodes, each storage node is initialized with a unique index. In the storage node, each received packet will be granted a timestamp. When a processing node retrieves data, the corresponding storage nodes will attach their indexes to relevant stored packets before sending. When a processing node receives a packet from a storage node, it can find the source of the packet by checking the index of the packet. In addition, due to transmission latency, the receiving sequence can be different than the sent sequence. To recover the original sequence, the processing node will sort the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2018.2859307, IEEE Transactions on Network Science and Engineering



(a) Transmission without Fault Tolerance

(b) Transmission after Fault Tolerance

Fig. 3: Comparison between Original and Improved Transmission

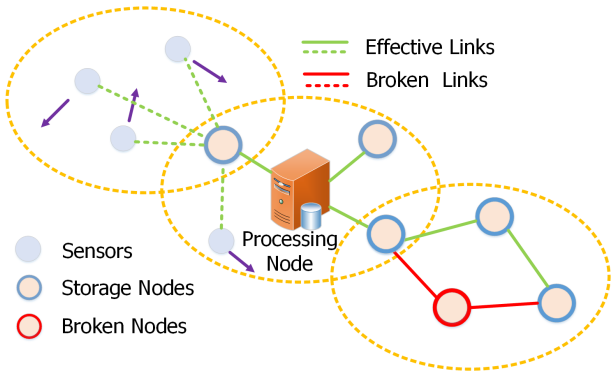


Fig. 2: System Model

received data based on timestamps.

Leveraging the trace back strategy above, a processing node is able to identify those lost packets. When sorting the received packets using timestamps, it can find the missing storage node indexes through a index table initialized during firing. Then, for each lost index, a corresponding $interest$ containing the index and timestamp is generated and broadcasted.

Once an $interest$ is received by a storage node, it will search whether it has the lost packet in its storage. If the corresponding packet is not stored in the storage, the broadcasting process will continue. On the other hand, if the packet is found, lost packets will be resent through limited flooding. Specifically, each storage node adopts the flooding algorithm to relay packets in a geographical rectangle using the point of the storage node and the processing node as vertices or a predefined logical range.

Both frequent broadcasting and flooding mechanisms can cause congestion, especially near the processing nodes. Assuming that the storage nodes near a processing node in a fog computing supported healthcare IoT system are more or less evenly distributed, we construct a graph $G = (V, E)$ whose nodes are distributed evenly, where V represents the set of nodes, and E represents the set of edges. Then, the average number of relays of an $interest$ at each node $\overline{\mathcal{R}_\mathcal{I}}$ should be

$$\overline{\mathcal{R}_\mathcal{I}} = \frac{|E|}{|V|}. \tag{1}$$

Let $\mathcal{D}_\mathcal{T}$ denote total data volume, $\mathcal{D}_\mathcal{R}$ denote the received data volume. Then, in the whole system, the maximum number of $interest$ $\mathcal{N}_\mathcal{I}$ is

$$\mathcal{N}_\mathcal{I} = \overline{\mathcal{R}_\mathcal{I}} \cdot |V| \cdot (\mathcal{D}_\mathcal{T} - \mathcal{D}_\mathcal{R}) = |E| \cdot (\mathcal{D}_\mathcal{T} - \mathcal{D}_\mathcal{R}). \tag{2}$$

Based on (2), the cost of broadcasting increases with the intensity of the graph. For the storage nodes, they have to relay packets that match with $interest$. Suppose that there is a processing node at point $(0, 0)$, and a specific storage node at point (x, y). Denote $r \in (0, R]$ as the distance between the processing node and the storage node where R is the maximum distance between the processing node and system boundary. Then, we have $r^2 = x^2 + y^2$. Accordingly, the flooding area of a storage node $\mathscr{A}$ should be

$$\mathscr{A} = \frac{|xy|}{\pi R^2}. \tag{3}$$

Since nodes are distributed approximately evenly, the number of nodes could be represented directly by its area. Furthermore, for each interested packet, the number of relays before it arrives at server $\mathcal{N}_{\mathcal{PT}}(x, y) \rightarrow svr$ is

$$\mathcal{N}_{\mathcal{PT}}(x, y) \rightarrow svr = \overline{\mathcal{R}_\mathcal{I}} \cdot |V| \cdot \mathscr{A} = \frac{|xyE|}{|\pi R^2|}. \tag{4}$$

Based on (4), the number of relays for packets sent from the storage nodes on a circle of radius r to the processing node $\mathcal{N}_{\mathcal{PT}}(r) \rightarrow svr$ can be calculated. To ensure the existence of nodes on this circle, we set a parameter δr, denoting the longest hop in the system. Then, all the storage nodes can be divided into $\frac{R}{\delta r}$ rings based on their distance from the processing node

$$\begin{aligned} r_1 &= \delta r \\ r_2 &= r_1 + \delta r \\ &\cdots \\ r_n &= r_{n-1} + \delta r \\ &\cdots \\ r_{\frac{R}{\delta r}} &= r_{\frac{R}{\delta r}-1} + \delta r. \end{aligned} \tag{5}$$

In (5), we call each ring area a field. For example, the area between r_{n-1} and r_n is called field n. Clearly, the longest possible distance between the storage nodes in field n and the processing node is $n \cdot \delta r$. Thus, we have

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2018.2859307, IEEE Transactions on Network Science and Engineering

$$\mathcal{N}_{\mathcal{PT}}(r) \rightarrow svr = \frac{\pi(r_n+\delta r)^2 - \pi r_n^2}{\pi R^2} \cdot |V| \cdot \frac{|xy|}{\pi R^2} \cdot |V| \cdot \frac{|E|}{|V|} \cdot \mathcal{FR}, \tag{6}$$

where $\frac{\pi(r_n+\delta r)^2-\pi r_n^2}{\pi R^2} \cdot |V|$ denotes the number of nodes in field n, $\frac{|xy|}{\pi R^2} \cdot |V| \cdot \frac{|E|}{|V|}$ is $\mathcal{N}_{\mathcal{PT}}(x,y) \rightarrow svr$, and $\mathcal{FR}$ denotes the failure rate.

As continuous broadcasting of interests results in the increase of resource occupancy, the cost grows with the distance between the nodes and the server. Accordingly, newly generated packets cannot get enough resources so that the overall quality may decrease. On the other hand, system resources are not fully used if the mechanism is excessively restricted. Thereby, we need to find a proper intermediary to allocate system resources properly. In the following section, a self-adaptation module is designed to achieve this target.

5 Self-adaptive Filtering

5.1 Filter

The transmission mechanism above ensures data completeness, but the data may be modified due to outside disruptive factors during transmission. We design a filter to address this issue, by checking the received data type and range before processing. Fig. 4 shows the structure of the filter. To provide the proper range of collected data, we introduce relevant data sets as a reference. A relevant data set contains historical data gathered by a healthcare IoT system, and is updated by a specific eHealth client periodically. Different eHealth clients usually hold their own business, for example, some focus on food and diet, some monitor exercise, and some pay more attentions to physiological status. They need to maintain their own relevant data sets to support their business. At the same time, the maintaining method keeps the relevant data sets to be as simple as possible.

For a received packet, the filter first extracts useful information from the packet header and body. Then, the filter will launch a category checking by identifying whether the data is health-related or not. Due to the fact that a storage node does not have a powerful processor, it may provide some data not related to health, leading to an error when processing. After that, the filter will check the data based on the range provided by corresponding relevant data sets. If the received data is out of the range from a relevant data set, the data will be regarded as invalid. To speed up the checking process, we adopt a learning machine to record common errors, and detect them faster.

The detected invalid data is recollected through reliable transmission, where data is recollected based on the indexes and the timestamps.

5.2 Self-adaptation Module

The fault-tolerant mechanism and the filter are designed for reliability. However, when transmission conditions are poor between the storage nodes and the processing nodes, the successful transmission ratio may degrade because of the high overhead. Therefore, we design a self-adaptation module to allocate system resources for the fault-tolerant mechanism according to the successful transmission ratio.

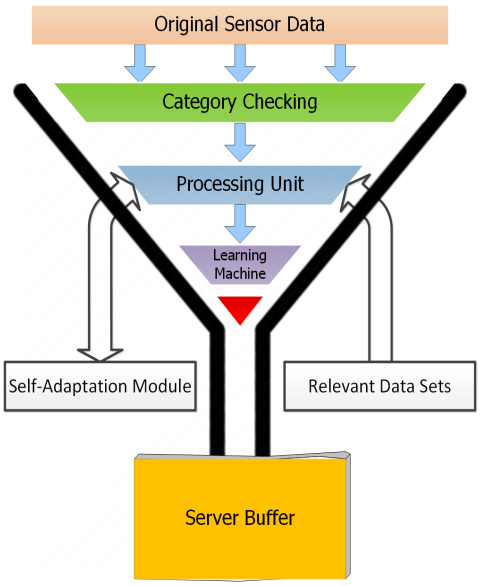


Fig. 4: Self-adaptive Filter

Some inaccurate data needs recollection after filtering which may decreases the valid data volume. Recall that we defined $\mathcal{D}_{\mathcal{T}}$ to represent total data volume, and $\mathcal{D}_{\mathcal{R}}$ for received data volume. Now, we define $\mathcal{D}_{\mathcal{C}}$ to indicate correct data volume out of what is received. Then, we have $\mathcal{D}_{\mathcal{T}} \geq \mathcal{D}_{\mathcal{R}} \geq \mathcal{D}_{\mathcal{C}}$, and according to D_C, the failure ratio $\mathcal{FR}$ can be defined as

$$\mathcal{FR} = \frac{\mathcal{D}_{\mathcal{T}} - \mathcal{D}_{\mathcal{C}}}{\mathcal{D}_{\mathcal{T}}}. \tag{7}$$

Meanwhile, (2) can be rewritten as

$$\begin{aligned} \mathcal{N}_{\mathcal{I}} &= \overline{\mathcal{R}_{\mathcal{I}}} \cdot |V| \cdot (\mathcal{D}_{\mathcal{T}} - \mathcal{D}_{\mathcal{C}}) \\ &= |E| \cdot (\mathcal{D}_{\mathcal{T}} - \mathcal{D}_{\mathcal{C}}) \\ &= |E| \cdot \mathcal{D}_{\mathcal{T}} \cdot \mathcal{FR}. \end{aligned} \tag{8}$$

The higher $\mathcal{FR}$ can be caused by decreased successful transmission ratio, which indicates worse transmission conditions. In this case, we should control resource usage strictly. Conversely, when $\mathcal{FR}$ becomes lower, the conditions are better. In this case, one can allocate more resources for transmissions.

To restrict the resource usage between storage nodes and processing nodes, two solutions are proposed in this section. One is to set a time to live (TTL) $\mathcal{T}_{\mathcal{L}}$ indicator for each $interest$. $\mathcal{T}_{\mathcal{L}}$ decreases by one after every hop relay, and the $interest$ disappears when $\mathcal{T}_{\mathcal{L}}$ becomes zero. The other approach is to set a maximum effective time (TOE) $\mathcal{T}_{\mathcal{E}}$ for each $interest$. $\mathcal{T}_{\mathcal{E}}$ increases by one for every successful match until TOE equals to its upper limit.

In the first solution, the total amount of $interest$ is fixed even in the worst case, due to the existence of $\mathcal{T}_{\mathcal{L}}$. In other words, every $interest$ disappears when it reaches $\mathcal{T}_{\mathcal{L}}$. Under this circumstance, (8) becomes

$$\mathcal{N}_{\mathcal{I}} = \mathcal{T}_{\mathcal{L}} \cdot \mathcal{D}_{\mathcal{T}} \cdot \mathcal{FR}. \tag{9}$$

Since $\mathcal{T}_{\mathcal{L}}$ is always less than $|E|$ (If $\mathcal{T}_{\mathcal{L}} \geq |E|$, a Depth-First-Search [24] will be launched for every $interest$), the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2018.2859307, IEEE Transactions on Network Science and Engineering

amount of $interest$ has been optimized. However, due to the existence of $\mathcal{T}_{\mathcal{L}}$, layers lower than $\mathcal{T}_{\mathcal{L}}$ in Breadth-First-Search [25] Tree will not be visited by any $interest$ if a Breadth-First-Search is launched from a processing node.

The second solution gives an $interest$ a chance to visit all the nodes. When $interests$ are not matched, they will never disappear, but broadcast continuously. However, $\mathcal{T}_{\mathcal{E}}$ cannot be reached by every $interest$, which leads to infinite broadcasting.

Our module takes the advantage of these two solutions. Let us define a new parameter $\mathcal{T}'_{\mathcal{E}}$. For an $interest$, if $\mathcal{T}_{\mathcal{L}}$ reaches its limit but effective time is less than $\mathcal{T}'_{\mathcal{E}}$, the $interest$ will continue transmission until TOE is reached. On the contrary, if $\mathcal{T}'_{\mathcal{E}}$ has been reached, the $interest$ disappears with $\mathcal{T}_{\mathcal{L}}$. Due to the adoption of $\mathcal{T}'_{\mathcal{E}}$, $\mathcal{T}_{\mathcal{L}}$ can be much smaller. According to equation (9), unnecessary relay of $interest$ can be reduced.

Another functionality of $\mathcal{T}'_{\mathcal{E}}$ is to avoid excessive resource occupancy by adjusting parameters dynamically, according to current condition. If the $\mathcal{T}'_{\mathcal{E}}$ is a fixed value, the infinite broadcasting problem is not solved. However, when we vary it based on network condition, the number of broadcasted $interests$ can be controlled gradually. Specifically, bandwidth between the storage nodes and the processing nodes is denoted as C, and capacity occupied by original transmission as C_0. As C_0 is relatively fixed, the remaining resource C_l that can be allocated by the fault-tolerant mechanism is

$$C_l = C - C_0. \tag{10}$$

Meanwhile, based on (6), $\mathcal{N}_{\mathcal{PT}}(r)$ denotes the total number of packets in field n, i.e.,

$$\mathcal{N}_{\mathcal{PT}}(r) = \sum_{r=r_{n+1}}^{R} \mathcal{N}_{\mathcal{PT}}(r) \rightarrow svr \cdot \frac{\pi(r_n+\delta r)^2 - \pi r_n^2}{\pi r^2}. \tag{11}$$

It can be calculated by accumulating the area proposition of field n in an area where $r \geq r_n$.

Then, (10) can be rewritten as

$$C_l = \mathcal{N}_{\mathcal{I}} + \sum_{r_n=0}^{R} \mathcal{N}_{\mathcal{PT}}(r_n). \tag{12}$$

Because $\mathcal{FR}$ exists in both $\mathcal{N}_{\mathcal{I}}$ and $\mathcal{N}_{\mathcal{PT}}(r_n)$, the mechanism will take up more resources when $\mathcal{FR}$ is higher. In this case, more $interests$ are generated and more bandwidth should be allocated to C_l, which may cause congestions. If we try to address this issue by decreasing $\mathcal{T}'_{\mathcal{E}}$, we note that more $interests$ will disappear with $\mathcal{T}_{\mathcal{L}}$. When the number of $interests$ is less, there are less nodes to response these $interests$, namely, $\frac{\pi(r_n+\delta r)^2-\pi r_n^2}{\pi R^2} \cdot |V|$ in (6) becomes smaller. As $|V|$ is a static value, the former part will be smaller. In (12), to keep C_l unchanged, we need to decrease $\mathcal{N}_{\mathcal{PT}}(r_n)$, which is possible because $\frac{\pi(r_n+\delta r)^2-\pi r_n^2}{\pi R^2}$ decreases. Accordingly, when $\mathcal{FR}$ increases, $\mathcal{T}'_{\mathcal{E}}$ needs to be reduced to keep the transmission reliable. Intuitively, $\mathcal{T}'_{\mathcal{E}}$ controls the number of $interests$ broadcasted in network. It increases with good network condition and more $interests$ are relayed while it decreases with poor condition, which reduces the number of $interests$ in network. The pseudo-code of the self-adaptation module is presented in ALGORITHM 1.

ALGORITHM 1: Self-Adaptation Module

Inputs: C_l, $\mathcal{FR}$

Output: $\mathcal{T}'_{\mathcal{E}}$

Procedure: Function $\mathcal{T}'_{\mathcal{E}}$`Decider()`

```
1 Begin
2   FR ← (D_T − D_C) / D_T
3   while (FR > 0)
4     if (C_l < N_I + Σ_{r_n=0}^{R} N_PT(r_n))
5       T'_E − −
6     else
7       T'_E + +
8     end if
9   end while
```

5.3 Level division

After checked by the filter, Data received by a processing node forms solution space, denoted as $\mathscr{S}$. The format of $\mathscr{S}$ can be expressed as the following matrix

$$\mathscr{S} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & x_{i3} & \cdots & x_{in} \end{bmatrix}, \tag{13}$$

where $1, 2, 3, ..., i$ and $1, 2, 3, ..., n$ represent data gathered from i storage nodes with same timestamp, and data gathered from one storage node at n different times, respectively. Afterwards, based on the boundary provided by the relevant data set, data in $\mathscr{S}$ will be set into a level structure consisting of several levels. For each division, we deal with a column of the solution space. In the column, we only need to randomly choose a row to divide it into a level, and then assign other data with same timestamp into the level. Furthermore, because the whole solution space serves as the input of RCA, the precision will remain the same even though we select different rows of the solution space during the division process. From a relevant data set, the processing node can retrieve the valid maximal and minimal values of h^{th} row, denoted as $x(h)_{max}$ and $x(h)_{min}$ $(h \in [1, i])$, respectively. Given a highest level $\mathscr{L}(max)$, the boundary of each level $\mathscr{L}(a)$ $(a \in [1, max])$ can be calculated as

$$\begin{aligned} \mathscr{L}(a)_{min} &= (a-1) \cdot \frac{x(h)_{max} - x(h)_{min}}{max} + 1 \\ \mathscr{L}(a)_{max} &= a \cdot \frac{x(h)_{max} - x(h)_{min}}{max}. \end{aligned} \tag{14}$$

Finally, each column in $\mathscr{S}$ belongs to a corresponding level, which will be called when constructing neighborhoods.

6 Data-load-reduction Processing

The expansion of healthcare IoT system will generate a large amount of data. Also, further development in healthcare IoT

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2018.2859307, IEEE Transactions on Network Science and Engineering

systems will provide diversified services. Based on these observations, an efficient and flexible interface is necessary between fog devices and eHealth clients. In this section, we introduce the risk assessor, and design a data-load-reduce scheme to help eHealth clients leverage sensor data computing services in a more friendly manner.

6.1 Risk Assessor

Risk assessor is a function returning a quantitative result used to identify the elderly person's level of danger under certain environments and physiological status. In REDPF, risk assessor is a utility function of RCA. RCA executes the risk assessor based on the input data to return refined results. The returned value is a threshold called risk factor, denoted by $\mathscr{R}$. Larger risk factors represent more dangerous situations. To notify eHealth clients about any potential dangers, an alarm will be sent if the factor exceeds a predefined value. We use risk threshold $\mathscr{R}_{\mathcal{TH}}$ to represent the predefined value. For example, if $\mathscr{R}$ varies between 0 and 100, and $\mathscr{R}_{\mathcal{TH}}$ is predefined as 80, eHealth clients can believe that an elderly person is healthy if the output of the risk assessment is less than 80. If the output is higher, an informative alarm will be sent to the clients regarding the individual's alarming status.

As eHealth clients run quite different businesses, risk assessors may vary a lot. Based on this consideration, fog computing providers will not define a risk assessor, but leave it as an interface to eHealth clients. With the interface, each eHealth client can design their own risk assessor, and retrieve what they need from the fog computing.

However, the freedom is at the price of efficiency. As eHealth clients are health care providers, they will consider less about the data amount, but more about the accuracy and results. Additionally, fog computing providers can hardly expect eHealth clients to design efficient algorithms with lowest complexity. Based on the above considerations, a suitable algorithm is indispensable when analyzing sensor data. In this paper, we take the advantage of the ideas of randomness and segmentations from RVNS, and design an optimized scheme, RCA, to improve the efficiency and accuracy.

6.2 RVNS-based Computing and Analyzing

When level division is ready, RCA first forms an initial neighborhood structure based on the current global optimal solution. At the beginning, RCA randomly generates an initial global optimal solution from $\mathscr{S}$ (a column of the matrix in (13)), denoted as X, and regards it as the current global optimal solution. Using the level of X, RCA forms a new neighborhood structure, $N(X)$. During the process, each column in the matrix is assigned in a neighborhood. Assume $X \subseteq \mathscr{L}(X)$ $(X \in [1, max])$, and another column belongs to level $\mathscr{L}(cur)$ $(cur \in [1, max])$, the column will be assigned into neighborhood $N_k(X)$, where $k = |\mathscr{L}(cur) - \mathscr{L}(X)|$.

When neighborhood construction is done, the requested data is in an RVNS queue. X enters the risk assessor, which will return a corresponding risk factor. In the following RVNS turns, RCA calls function $Shake()$ to randomly select a new local optimal solution X' from neighborhoods. The process starts with the first neighborhood. If the newly generated risk factor of X' exceeds that of X, the local optimal solution X' will replace X to become the new global optimal solution, based on which, a new neighborhood structure will be generated. If a risk factor exceeds the risk threshold, an alarm will be sent to eHealth clients. Conversely, if the original risk factor from current global optimal solution X is larger, RCA will switch to next neighborhood, and continue the searching process. The pseudo-code of RCA scheme is presented in ALGORITHM 2.

ALGORITHM 2: RCA

Inputs: $\mathscr{S}$, $\mathscr{R}_{\mathcal{TH}}$, max

Output: A Boolean variable for sending alarms

Procedure: Function `RCA()`

```
1  Begin
2    X ←RCAInitialize()
3    while (R < R_TH)
4      NeighborhoodConstruct(X)
5      k = 1
6      while (k < (max − 1))
7        X' ←Shake(X, k)
8        if (f(X') > f(X))
9          X ← X'
10         break
11       else
12         k ← k + 1
13       end if
14     end while
15   end while
16   report()
```

6.3 Detailed Design of RCA

During the implementation of RCA, the performance of the scheme may be heavily influenced by two factors. The size of the solution space becomes one of them. This is because RCA picks data from RVNS queue randomly, the possibility of finding the risk data is reducing when the solution space becomes larger. Conversely, when the solution space is small enough, RVNS queue can be approximated as a FIFO queue. The other important factor influencing the performance is the interval of neighborhood reconstruction. As a processing node receives data continuously, latest data has to wait for next RVNS turn if the current turn has begun. The larger the interval is, the longer will be the waiting time. However, the processing node may not have enough time to analyze current solution space for short intervals. To include some potential cross-impacts of these two factors, we perform extensive simulations to find their optimal values.

Finally, we modify the terminal condition of RVNS to better meet the requirements of healthcare IoT systems. Originally, designers of RVNS algorithm proposed two terminal conditions: neighborhood reconstructing times and CPU time. For the first condition, RVNS stops once the time of neighborhood reconstruction reaches the predefined value.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2018.2859307, IEEE Transactions on Network Science and Engineering

TABLE 1: Configuration for the Reliability Experiment

Scene Parameters	Simulation Time (s)	6000
	System Scale (m^2)	100×100
	Compare Algorithms	Original Routing, Interests-based Routing, Self-adaptation Routing
Nodes Parameters	Number of Storage /Trunk Nodes	100
	Number of Failure Nodes	10, 20, 30, 40, 50
	Number of Processing Nodes	1

It is not suitable for our case, because when RCA stops, it caches a relatively large risk factor. However, restarting of the whole scheme means discarding the larger factor and using a smaller one, which further increases the processing time. The same problem will still appear if the second condition is adopted. Under the circumstance, it is notable that when the system sends an alarm to eHealth clients, the risk factor at that time is larger than at any other time. Afterwards, RCA will not stop for any other smaller factors, even if they exceed the threshold. In this case, eHealth clients may pay limited attention to some potential dangers. Based on this consideration, we restart the whole scheme when an alarm is sent.

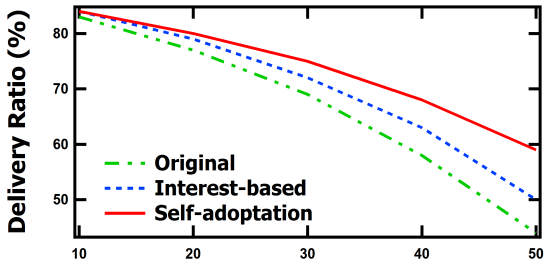
7 Numerical Results

We execute simulation experiments to evaluate the proposed framework in terms of reliability and processing performance using C++. We try to simulate a fog computing supported healthcare IoT system in a 100×100 m^2 two-dimensional space. In the simulation environment, we set 100 storage or trunk nodes, and to avoid synchronizing issues, we deploy only one processing node. We simulate packets delivery processes within the above environment in order to observe different performance in the original model, the interest-based model, and self-adaptation delivery models. In the second part, we compare the processing speed between RCA and traditional FIFO. In this part, the processing node runs the RCA and FIFO programs automatically with input data, and the programs return the processing situation and risks to eHealth clients.

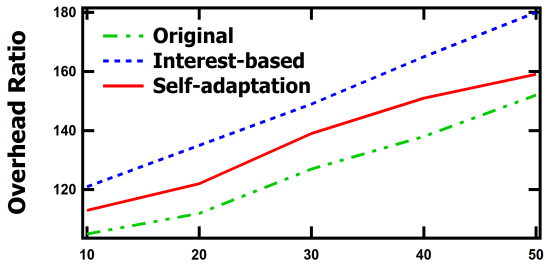
7.1 Reliability Performance

We evaluate the performance of reliability under different failure conditions. The parameters used for this section is presented in Table 1. In the experiment, totally 101 nodes (100 storage/trunk nodes and one processing node) are deployed in a $100 \times 100m^2$ area. The processing node will send virtual requests towards different storage nodes continuously. Once a storage node receives the request, it executes corresponding models to respond it, which simulates the process of packets delivery. Then, we observe the change in the delivery ratio at the processing node, and the overhead ratio during the transmission. To demonstrate the fault tolerant ability, we set number of failure storage/trunk nodes to be 10, 20, 30, 40 or 50. The simulation results are shown in Fig. 5.

As shown in Fig. 5, the interest-based delivery can optimize the performance to some extent, but due to its own deficiency, the overhead ratio also increases dramatically, which becomes a limitation. The self-adaptation delivery solves the problem above. As shown in Fig. 5(b), under the optimized scenario, the delivery ratio improves further while keeping the overhead ratio lower than in the case of interest-based delivery.



(a) Delivery Ratio



(b) Overhead Ratio

Fig. 5: (a): Delivery Ratio. (b): Overhead Ratio

TABLE 2: Configuration in Processing Performance Experiment

Scene Parameters	Simulation Time (s)	50000
	Number of Levels	4
	Risk Factor Range	[0,100]
	Risk Factor Threshold	90
	Risk Emerging Time (s)	5000,7500,10000, 12500,15000,17500, 20000,22500,25000
	Risk Emerging Probability	0.05,0.1,0.15,0.2,0.25, 0.3, 0.35, 0.4, 0.45, 0.5
Server Parameters	Data Receiving Interval (s)	5
	Group of Data in Solution Space	100, 200, 300, 400, 500, 600, 700, 800, 900, 1000
	Interval of Neighbo-rhood Reconstruction	100, 200, 300, 400, 500, 600, 700, 800, 900, 1000
Data Parameters	Average Processing Time (s) (for scenario 3)	5, 6, 7, 8, 9, 10, 11,12,13,14,15
	Processing Time Range (s) (for other scenarios)	[5,12]

7.2 Data Processing Performance

RCA is evaluated through the second part of the simulation in this section. The simulation can be divided into two phases. The first phase finds the optimal parameters of RCA. The second phase compares the running speed of RCA with traditional FIFO queue.

The overall setting is demonstrated in Table 2. In our simulation, a processing node to monitor heart rate abnormalities is set up. Arrhythmia Data Set from UCI Machine Learning Repository [26] is served as the data source. In

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2018.2859307, IEEE Transactions on Network Science and Engineering

the simulation, each group of input enters the processing node per five seconds, and the input is stored in a FIFO queue and an RVNS queue constructed from four levels, respectively. To return the risk factor, we adopt an black box to encapsulate risk assessor for data processing. For each group of input, the processing node takes some time to analyze, ranging from 5 to 12s (integer). The maximum risk factor is set as 100, and the risk threshold $\mathscr{R}_{\mathcal{TH}}$ is set as 90. The experiment lasts 50000s.

In the experiment, we first observe the time needed to return risk data when the size of the solution space is changing with the interval of neighborhood reconstruction. The results are presented in Fig. 6, where values of both factors are 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000. By doing the grid search, we are aiming to seek a pair of optimal parameters for RCA.

Afterwards, we take the optimal parameters, and observe the change of maximal recorded risk factors with time for both RCA and FIFO. The input data stream for both methods is exactly the same. As described earlier, both RCA and FIFO are going to run through the risk assessor, and keep the max risk factor. The purpose of this setup is essentially to plot a memory variable for RCA and FIFO. A higher risk factor at the same moment represents the corresponding method has the ability to find larger risk factor more rapidly. The results are presented in Fig. 7.

In the end, we set the following scenes, and observe the waiting time of RCA and FIFO. (1) Risk data appears after 5000, 7500, 10000, 12500, 15000, 17500, 20000, 22500, 25000 seconds; (2) Possibilities of risk appearance are 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5; (3) Average processing time of each group of data are 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 s. As described earlier, RCA will run through the risk assessor and find and keep the max risk factor. Since last comparison illustrates the ability of finding larger risk factor in a specific situation, we try to expand to more situations. Accordingly, we set above three groups of experiment to compare the performance of RCA and FIFO more extensively. Because we are going to set risk appearing time (s) as one of the parameters, we decide to use waiting time (s) after the risk appearing as our performance metric, which will be more intuitive. The results are presented in Fig. 8.

To eliminate cross impact, we change both parameters in Fig. 6 together. As shown in the results, the optimum of both parameters are 100, so the value will be adopted for rest of the evaluation.

As shown in Fig. 7 and Fig. 8, once the data arriving ratio is larger than the processing ratio, RCA provides a quicker response in all cases because RVNS queue changes data processing sequence. The demonstrated processing efficiency reveals that the potential risk can be detected and reported earlier.

7.3 Discussions

To summarize the simulation, we can conclude that the fault-tolerant mechanism and self-adaptation module optimize the transmission process in fog computing supported healthcare IoT system. It provides insurance for the later processing. Then, we find that RCA is highly efficient for big data analysis when the capability of the processor is limited. In addition, the framework reduces the data load transmitted to eHealth clients significantly.

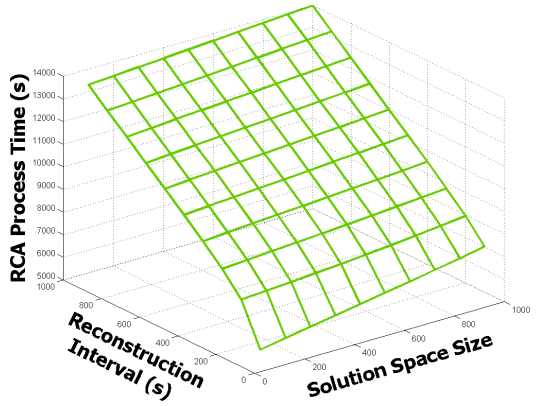


Fig. 6: RCA Process Time (Risks emerge after 5000s; the probability of risk is 0.1).

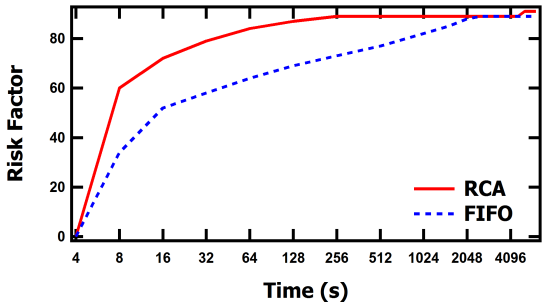


Fig. 7: Highest founded risk factor changing with time (Risks emerge after 5000s; the probability of risk is 0.1).

8 Conclusion

We have proposed a framework for fog computing supported healthcare IoT system. In the framework, we have proposed a fault-tolerant mechanism by combining the advantages of Directed Diffusion and Limited Flooding to enhance the reliability of data transmission. In addition, a self-adaptation Module has been designed to allocate resources inside the system to avoid overuse. Finally, we have proposed an RVNS queue to process filtered data. In the Queue, the processor has the chance to access the latest received data quickly to enhance processing speed. According to the simulation, we have first proved that the fault-tolerant mechanism and the self-adaptation module can improve the successfully delivered ratio as well as optimize the resource allocation. Then, by comparing the performance of RVNS queue and FIFO queue in different scenarios, we claim that RCA is a competitive scheme for big data processing in fog computing supported healthcare IoT system.

References

[1] A. K. M. Azad, J. Kamruzzaman, B. Srinivasan, K. M. Alam, and S. Pervin, "Query Processing over Distributed Heterogeneous Sensor Networks in Future Internet: Scalable Architecture and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2018.2859307, IEEE Transactions on Network Science and Engineering

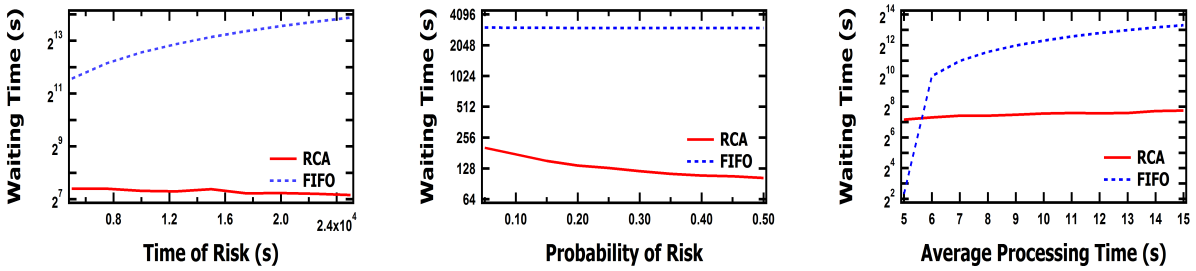


Fig. 8: (a): Waiting time of data changing with risks emerging time (The probability of risk is 0.1). (b): Waiting time changing with probability of risks emerging (Risks emerge after 5000s). (c): Waiting time changing with average data process time (Risks emerge after 5000s; the probability of risk is 0.1).

Challenges," *2010 Second International Conference on Advances in Future Internet*, 2010, pp. 75-81.

[2] P. Hansen and N. Mladenovic, "Variable neighborhood search: Principles and applications," *European Journal of Operational Research*, vol. 130, no. 3, pp. 449-467, 2001.

[3] C. Min and Y. Eom. "Integrating Lock-Free and Combining Techniques for a Practical and Scalable FIFO Queue," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 7, pp. 1910-1922, 2015.

[4] G. D. Pietro and A. Coronato, "Tools for the Rapid Prototyping of Provably Correct Ambient Intelligence Applications," *IEEE Transactions on Software Engineering*, vol. 38, no. 4,pp. 975-991, 2012.

[5] R. Ssembatya and A. V. D. M. Kayem, "Secure and Efficient Mobile Personal Health Data Sharing in Resource Constrained Environments," *IEEE International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pp. 411-416, 2015.

[6] J. Winkley, P. Jiang, and W. Jiang, "Verity: an ambient assisted living platform," *IEEE Transactions on Consumer Electronics*, vol. 58, no. 2, pp. 364-373, 2012.

[7] T. Magherini, A. Fantechi, C. D. Nugent, and E. Vicario, "Using temporal logic and model checking in automated recognition of human activities for ambient-assisted living," *IEEE Transactions on Human-Machine Systems*, vol. 43, no.6, pp. 509-521, 2013.

[8] J. C. Wang, C. H. Lin, E. Siahaan, B. Chen, and H. Chuang, "Mixed sound event verification on wireless sensor network for home automation," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, pp. 803-812, 2014.

[9] B. Y. Xu, L. D. Xu, H. M. Cai, C. Xie, J. Hu, and F. Bu, "Ubiquitious data accessing method in IoT-based information system for emergency medical services," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1578-1586, 2014.

[10] C. H. Liu, J. Wen, Q. Yu, B. Yang, and W. Wang, "HealthKiosk: A family-based connected healthcare system for long-term monitoring," *IEEE INFOCOM*, pp. 241-246, 2011.

[11] M. Barua, X. Liang, R. Lu, and X. Shen, "PEACE: An efficient and secure patient-centric access control scheme for eHealth care system," *IEEE INFOCOM*, pp. 879-886, 2010.

[12] A. Bamis, D. Lymberopoulos, T. Teixeira, and A. Savvides, "The BehaviorScope framework for enabling ambient assisted living," *Personal and Ubiquitous Computing*, vol. 14, no. 6, pp. 473-487, 2010.

[13] M. Ruta, F. Scioscia, G. Loseto, and E. D. Sciascio, "Semantic-based resource discovery and orchestration in Home and Building Automation: a multi-agent approach," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, pp. 730-741, 2014.

[14] H. Mei, B. J. Beijnum, I. Widya, V. Jones, and H. Hermens, "Enhancing the performance of mobile healthcare systems based on task-redistribution," *IEEE INFOCOM*, pp. 1-6, 2008.

[15] N. K. Suryadevara and S. C. Mukhopadhyay, "Determination of Wellness of an Elderly in an Ambient Assisted Living Environment," *IEEE Intelligent Systems*, vol. 29, no. 3, pp. 30-37, 2014.

[16] M. A. Hossain, P. K. Atrey, and A. E. Saddik, "Modeling and assessing quality of information in multisensor multimedia monitoring systems," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 7, no. 1, pp. 33-63 2011.

[17] L. Cheng, J. Niu, Y. Gu, C. Luo, and T. He, "Achieving Efficient Reliable Flooding in Low-Duty-Cycle Wireless Sensor Networks," *IEEE/ACM Transactions on Networking*, vol. 24, no. 6, pp. 3676-3689, 2016.

[18] F. Z. Benhamida, and Y. Challal, "FaT2D: Fault Tolerant Directed Diffusion for Wireless Sensor Networks," *IEEE International Conference on Availability, Reliability and Security*, pp. 112-118, 2010.

[19] S. K. Singh, M. P. Singh, and D. K. Singh, "A survey of energy-efficient hierarchical cluster-based routing in wireless sensor networks," *International Journal of Advanced Networking and Application*, vol. 2, no. 2, pp. 570-580, 2010.

[20] I. Stojmenovic and W. Sheng, "The fog computing paradigm: Scenarios and security issues," *IEEE Federated Conference on Computer Science and Information Systems*, pp. 1-8, 2014.

[21] Q. Hu, C. Wu, X. Zhao, X. Chen, Y. Ji, and T. Yoshinaga, "Vehicular Multi-access Edge Computing with licensed Sub-6 GHz, IEEE 802.11p and mmWave," *IEEE Access*, Vol. 6, no. 1, pp. 1995-2004, 2017.

[22] Y. Cao, S. Chen, P. Hou and D. Brown, "FAST: A fog computing assisted distributed analytics system to monitor fall for stroke mitigation," *IEEE International Conference on Networking, Architecture and Storage*, pp. 2-11, 2015.

[23] M. Aazam and E. Huh, "E-HAMC: Leveraging Fog computing for emergency alert service," *IEEE International Conference on Pervasive Computing and Communication Workshops*, pp. 518-523, 2015.

[24] M. S. Mahmud, U. Sarker, M. M. Islam, and H. Sarwar, "A Greedy Approach in Path Selection for DFS Based Maze-map Discovery Algorithm for an autonomous robot," *IEEE International Conference on Computer and Information Technology*, pp. 546-550, 2012.

[25] F. Busato and N. Bombieri, "BFS-4K: an Efficient Implementation of BFS for Kepler GPU Architectures," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 7, pp. 1826-1838, 2015.

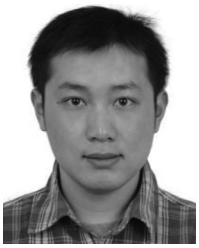
[26] H. A. Guvenir, B. Acar, and H. Muderrisoglu, "Arrhythmia Data Set in UCI Machine Learning Repository," http://archive.ics.uci.edu/ml/datasets/Arrhythmia.

Kun Wang (M'13-SM'17) received two Ph.D. degrees from Nanjing University of Posts and Telecommunications, China in 2009 and from the University of Aizu, Japan in 2018, respectively, both in Computer Science. From 2013 to 2015, he was a Postdoc Fellow in Electrical Engineering Department, University of California, Los Angeles (UCLA), CA, USA. He is currently a Research Fellow in the Department of Computing, the Hong Kong Polytechnic University, Hong Kong, China, and also a Full Professor in the School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing, China. He has published over 100 papers in referred international conferences and journals. He has received Best Paper Award at IEEE GLOBECOM16. He serves as Associate Editor of IEEE Access, Editor of Journal of Network and Computer Applications, Journal of Communications and Information Networks, EAI Transactions on Industrial Networks and Intelligent Systems and Guest Editors of IEEE Access, Future Generation Computer Systems, Peer-to-Peer Networking and Applications, and Journal of Internet Technology. He was the symposium chair/co-chair of IEEE IECON16, IEEE EEEIC16, IEEE WCSP16, IEEE CNCC17, etc. He is a Senior Member of IEEE and Member of ACM.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2018.2859307, IEEE Transactions on Network Science and Engineering

Yun Shao is a postgraduate student in Viterbi School of Engineering, Department of Computer Science, University of Southern California (USC). He received his Bachelor degree at Nanjing University of Posts and Telecommunications, China. His current research interests include Wireless Sensor Network, Delay Tolerant Network and Stream Computing.

Song Guo (M'02-SM'11) is a Full Professor at Department of Computing, The Hong Kong Polytechnic University. He received his Ph.D. in computer science from University of Ottawa and was a professor with the University of Aizu. His research interests are mainly in the areas of big data, cloud computing and networking, and distributed systems with over 400 papers published in major conferences and journals. His work was recognized by the 2016 Annual Best of Computing: Notable Books and Articles in Computing in ACM Computing Reviews. He is the recipient of the 2017 IEEE Systems Journal Annual Best Paper Award and other five Best Paper Awards from IEEE/ACM conferences. Prof. Guo was an Associate Editor of IEEE Transactions on Parallel and Distributed Systems and an IEEE ComSoc Distinguished Lecturer. He is now on the editorial board of IEEE Transactions on Emerging Topics in Computing, IEEE Transactions on Sustainable Computing, IEEE Transactions on Green Communications and Networking, and IEEE Communications. Prof. Guo also served as General, TPC and Symposium Chair for numerous IEEE conferences. He currently serves as an officer for several IEEE ComSoc Technical Committees and a director in the ComSoc Board of Governors.



Lei Xie received the PhD degree in computer science from Nanjing University. He is an associate professor in the Department of Computer Science and Technology at Nanjing University. His research interests include RFID systems, pervasive and mobile computing, and internet of things. He has published more than 30 papers in IEEE Transactions on Parallel and Distributed Systems, ACM MobiHoc, IEEE INFOCOM, IEEE ICNP, IEEE ICC, IEEE GLOBECOM, MobiQuitous, etc. He is a member of the IEEE.

Jie Wu is the Associate Vice Provost for International Affairs at Temple University. He also serves as the Chair and Laura H. Carnell professor in the Department of Computer and Information Sciences. Prior to joining Tempe University, he was a program director at the National Science Foundation and was a distinguished professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. Dr. Wu regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE Transactions on Service Computing and the Journal of Parallel and Distributed Computing. Dr. Wu was general co-chair/chair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, and ACM MobiHoc 2014, as well as program co-chair for IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). Dr. Wu is a CCF Distinguished Speaker and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.