Computer Networks 194 (2021) 108125

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

TONTA: Trend-based Online Network Traffic Analysis in ad-hoc IoT networks

Amin Shahraki [a,b,*], Amir Taherkordi [a], Øystein Haugen [b]

[a] *Department of Informatics (IFI), University of Oslo, Oslo, Norway*
[b] *Faculty of Computer Sciences, Østfold University College, Halden, Norway*

ARTICLE INFO

Keywords:
Internet of Things
NTMA
Pattern recognition
Trend change detection
Network Traffic Analysis
Network monitoring

ABSTRACT

Internet of Things (IoT) refers to a system of interconnected heterogeneous smart devices communicating without human intervention. A significant portion of existing IoT networks is under the umbrella of ad-hoc and quasi ad-hoc networks. Ad-hoc based IoT networks suffer from the lack of resource-rich network infrastructures that are able to perform heavyweight network management tasks using, e.g. machine learning-based Network Traffic Monitoring and Analysis (NTMA) techniques. Designing light-weight NTMA techniques that do not need to be (re-) trained has received much attention due to the time complexity of the training phase. In this study, a novel pattern recognition method, called Trend-based Online Network Traffic Analysis (TONTA), is proposed for ad-hoc IoT networks to monitor network performance. The proposed method uses a statistical light-weight Trend Change Detection (TCD) method in an online manner. TONTA discovers predominant trends and recognizes abrupt or gradual time-series dataset changes to analyze the IoT network traffic. TONTA is then compared with RuLSIF as an offline benchmark TCD technique. The results show that TONTA detects approximately 60% less false positive alarms than RuLSIF.

1. Introduction

IoT is increasingly recognized as a networking paradigm connecting heterogeneous smart devices without human intervention [1,2]. The heterogeneity can be in different aspects such as energy, processing power, and communication protocols [3]. IoT can be established based on two types of network infrastructures including: (i) *Infrastructure-dependent* networks: Some fixed devices are in charge of connecting end devices, e.g. routers, switches, and Base Stations (BSs). and (ii) *Infrastructure-independent or ad-hoc* networks: No fixed node is available to establish the network infrastructure. Instead, some resource-rich end-nodes temporarily work together to establish the network infrastructure and connect other nodes.

A significant portion of real-world IoT networks is under the umbrella of ad-hoc networking. In several IoT applications, the nodes use conventional ad-hoc communication protocols, e.g. Zigbee, or use Device to Device (D2D)-enabled communication protocols, e.g. WiFi-Direct and Cellular-D2D [4,5]. Fig. 1 shows an example of ad-hoc IoT networks. Considering data forwarding tasks in ad-hoc IoT, existing techniques have been almost successful in establishing an efficient network infrastructure, e.g. routing and clustering techniques [6–8]. On the contrary, in case of NTMA, ad-hoc IoT is not mature yet [9].

Monitoring and analyzing the network traffic is a key requirement in such networks like many other conventional networks. NTMA is a broad topic, including sub-fields of traffic classification, traffic prediction, fault management, and network security, to name a few. Network Traffic Analysis (NTA) is a sub-field of network traffic prediction [10,11] which plays an important role in monitoring the network traffic behavior and efficiency. In addition, it can detect unusual events or gradual departures from a normal operation by analyzing network traffic flow characteristics. Generally, given the size and complexity of data and the dynamicity of the network, NTMA tasks are considered heavyweight.

In Infrastructure-dependent IoT networks, resource-rich devices, e.g. routers and switches are to perform heavy tasks of monitoring and analyzing the network traffic. On the contrary, in infrastructure-independent networks, most of the infrastructure nodes have not enough resources to perform such heavy NTMA tasks. Although existing NTMA solutions have good performance, most of them are based on Machine Learning (ML) techniques [9]. ML-based NTMA techniques suffer from some disadvantages in the case of ad-hoc IoT as indicated below [12]:

- Unlike other ML applications, in network management solutions, ML techniques should be retrained frequently due to the high dynamicity of networks. Some events can cause ML models retraining [13] due to daily training [14], changes in analysis requirements (by the NTMA platform itself or human) [15], changes in the network traffic behavior [16], and starting a new network traffic (the ML model should be trained from scratch). Most ML

* Corresponding author at: Department of Informatics (IFI), University of Oslo, Oslo, Norway.
E-mail address: am.shahraki@ieee.org (A. Shahraki).

https://doi.org/10.1016/j.comnet.2021.108125
Received 11 September 2020; Received in revised form 23 March 2021; Accepted 13 April 2021
Available online 17 April 2021

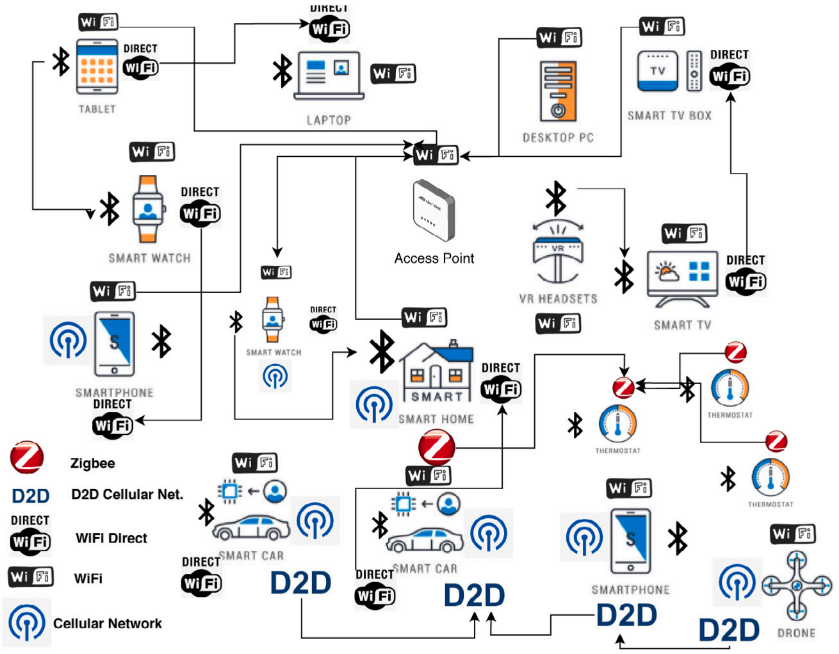

Fig. 1. A limited-size example of ad-hoc based IoT networks.

techniques need a considerable amount of resources to be trained. Training the ML models is also a time-consuming task that cannot support time-sensitive IoT applications.

- Conventional supervised learning techniques are efficient only in the case of labeled data, but in real-world applications of networking, most of the data is unlabeled or semi-labeled [17]. Deep Learning (DL) is the most well-known ML technique to work with unlabeled data [18], but it has some drawbacks, mainly it demands high resources and time to train the DL model.

Based on the above disadvantages, ML-based NTMA techniques are considered resource demanding for ad-hoc resource-constrained IoT. To monitor networking efficiency, we need to invest in more lightweight methods. Compared to ML techniques, statistical techniques are more lightweight NTMA methods that generally do not need to be trained. The network traffic characteristics, e.g. delay, throughput and jitter can be presented as time-series datasets as they are extracted gradually from the network in a period of time. In statistical techniques, pattern recognition methods are the most common solution to discover trends of time-series datasets [19,20]. Network traffic pattern recognition is a broad topic and includes some sub-techniques [21], e.g. anomaly detection and trend change detection. TCD is a solution generally designed for discovering trends of time-series datasets. TCD techniques also include various sub-techniques, e.g. time series segmentation and Change Point Detection (CPD) [22]. Among various TCD sub-techniques, CPD techniques are the most common online solutions to identify changes in gradually-generated time-series datasets [23]. CPD techniques process the time-series dataset online to detect Change points (CPs) (as data points) that can divide the datasets into subsets with predominant trends. Compared to ML techniques, CPD techniques are very light-weight [24]. In addition, they do not need to be (re-) trained.

We remedy the lack of lightweight NTA techniques by proposing a network traffic monitoring technique based on the online TCD technique. The proposed technique determines the behavior of IoT network traffic by analyzing network management characteristics gathered as time-series datasets. The proposed method, called TONTA, can be used by all nodes in charge of forwarding data, e.g. middle nodes and gateways. To analyze the network traffic based on TONTA, we consider two steps: (*i*) determining boundaries of network traffic behavior changes by recognizing CPs of the time-series dataset, and (*ii*) identifying the trends between every two continuous boundaries as the network traffic behavior. The methods introduced in [25,26] are the cornerstone of TONTA, but it includes fundamental improvements compared to the above works. The key contributions of this paper include:

- Proposing a new TCD method that does not need (re)training.
- Designing a NTA method to recognize abrupt and gradual changes in the network traffic in an online manner.
- Formulating a new model for a dynamic-sliding window to process datasets gathered from ad-hoc based IoT networks to improve the time complexity and accuracy of the proposed algorithm.

The rest of the paper is structured as follows. In Section 2, related work is discussed. In Section 3, we present TONTA, while its efficiency is evaluated and compared with Relative unconstrained Least-Squares Importance Fitting (RuLSIF) in Section 4. Finally, Section 5 concludes the paper and highlights the future directions. Table 1 shows the acronyms used in this paper.

2. Related work

There are some survey papers reviewing the NTMA techniques, e.g. [9,27]. To support QoS, the network infrastructure should be monitored periodically to detect abnormal or abrupt network traffic changes [28,29]. In terms of traffic prediction, both classic forecasting methods and machine learning techniques are introduced. As the statistical-based methods have lower overhead compared to ML techniques, they have attracted much attention in network monitoring. In [30], the authors evaluate the three types of predictors including classic time series, artificial neural networks and wavelet transform-based predictors through real network traces. They show that Double Exponential Smoothing (DES) as a classic time-series predictor has a good trade-off between performance and cost overhead. In [31], the authors have discussed about statistical methods for network surveillance from different aspects, e.g. network security, data network and dynamic networks.

In case of classic forecasting, Auto Regressive Integrated Moving Average (ARIMA) is one of the most well-known techniques [32]. In [33], Mehdi et al. introduce *FARIMA* based on combining the ARIMA and fuzzy regression to forecast traffic in cloud computing. They reduce

Table 1
List of abbreviations.

Abbreviation	Description
ANN	Artificial Neural Network
ARIMA	Auto Regressive Integrated Moving Average
BS	Base Station
CDM	Curve Detection Method
CP	Change Point
CPD	Change Point Detection
D2D	Device to Device
DES	Double Exponential Smoothing
DL	Deep Learning
DWT	Discrete Wavelet Transform
HIAD	Half Interaction Anomaly Degree
IoT	Internet of Things
ML	Machine Learning
NTA	Network Traffic Analysis
NTMA	Network Traffic Monitoring and Analysis
OWD	One-Way Delay
PLC	Probabilistic Link Capacity
QoS	Quality of Service
RNN	Recurrent Neural Network
RuLSIF	Relative unconstrained Least-Squares Importance Fitting
SPI	Shallow Packet Inspection
TCD	Trend Change Detection
TONTA	Trend-based Online Network Traffic Analysis

the need for historical data by ARIMA using the fuzzy regression and also propose SOFA as an sliding window to improve the accuracy of the prediction. The authors in [34] apply fractional ARIMA to predict the network throughput in adaptive video based on HTTP. In [35], the authors introduce a ARIMA-based modeling for non-Gaussian traffic data to manage traffic congestion by predicting abnormal status. They use R for data preprocessing and use the ARIMA to analyze and predict the traffic. In [36], Schemidt et al. introduce an unsupervised detection approach for cloud monitoring based on an online ARIMA prediction technique for online data monitoring to detect degraded state anomalies. Their results show that online ARIMA is efficient in case of anomaly detection with high accuracy and low number of false alarms. In [37], the authors introduce a network traffic prediction technique based on combining Discrete Wavelet Transform (DWT), ARIMA and Recurrent Neural Network (RNN). They use the proposed model to predict the network traffic based on time-series dataset to improve Quality of Service (QoS) and reduce cost.

Statistical based techniques are also used for supporting QoS. In [38], the authors use the first and second order statistics to propose a metric to predict amount of data that can be delivered through a shared band of a spectrum to guarantee QoS statistically. They introduce a probabilistic link capacity (PLC) metric based on a distributed robust data-driven approach. They use the estimation errors to evaluate the uncertainty of the statistics. In [39], the authors introduce a network monitoring system based on extensible SDN and NFV. Their proposed monitoring mechanism is based on a statistical data analysis technique by NetFlow to collect information through virtual routers.

Anomaly detection as an sub-field of NTMA is very popular for detecting abnormal changes in computer networks [40,41], but they are generally used to improve security of the network. Anomaly detection methods monitor the nodes and also traffic flows of the network to evaluate the efficiency [42,43]. Although anomaly detection methods are generally used to improve security, monitoring the network to improve QoS is another task which can be performed by them [44]. Flow-based anomaly detection methods determine the behavior of packet streams to detect abnormal or abrupt changes [45]. Several techniques are introduced to detect anomalies based on analyzing traffic flows in a network [46,47].

Although different statistical methods are used to improve the performance of networking, CPD techniques have attracted much attention. In [48], the authors proposed a model to transform traditional IP flow analyses to stream-based IP flow analysis to improve security and real-time situational awareness with high throughput, low latency, and good scalability. The authors in [49] introduce a model for continuous detection of performance anomalies based on the property of streams which has two levels. First, they use an adaptive learning model to estimate an underlying temporal property of the stream. Then, they use robust control charts based on statistics to recognize deviations. In [50], the authors use a non-parametric model to describe the congestion times during transferring a file using a CPD method. They use the proposed method to create time periods, dividing data into subsets and put each subset on the time periods called segments to analyze the subsets.

TCD is also used in anomaly detection techniques. In [51], the authors introduce a method to detect DDoS attacks based on extracting HIAD form IP based network flows and analyze it using trend change detection methods. In [52], M. Alkasassbeh proposed a new prediction model to estimate network traffic and use a CPD method to detect DDoS attacks in computer networks.

In [53], the authors introduce an anomaly detector method in networks based on time variations and correlation among statistics gathered from the network as time-series datasets. Table 2 compare the literature as reviewed in this section.

3. Proposed method

Different QoS metrics can be gathered from the network to collect *network traffic characteristics*, e.g. throughput, delay and jitter. We focus on *packet delay* (briefly delay) as an important metric in time-sensitive IoT applications. One-way Delay (OWD), also called end-to-end delay, is an important QoS metric to reveal network traffic behavior changes as a time-series dataset [54,55]. A time-series dataset follows various types of trends, but mainly, each of them can be uptrend, downtrend, or side-way trend. In OWD time-series datasets, uptrend shows the delay of the packet stream is increasing and QoS cannot be provided after a while. A downtrend shows the delay is decreasing and some forwarding resources will be free. A side-way trend shows the packet stream behavior is stable and no important change is happening. TONTA follows some steps to discover the network traffic behavior as listed below:

1. As the first-step of the data pre-processing, before each trigger, TONTA waits for enough new data points (i.e. OWD raw data) to perform a sampling method. After gathering enough raw OWD data, the sampling method is performed to calculate a data point, called sampled data point. TONTA appends the sampled data to the end of the time-series dataset. We consider that the OWD raw data is extracted by Shallow Packet Inspection (SPI), but performing SPI is out of the scope of this study. When enough sampled data is appended to the time-series dataset ($Interval\_Thr$), TONTA proceeds to the next step.
2. TONTA adapts the size of a dynamic-sliding window to select an adequate number of data points from the end of the time-series dataset.
3. As the second-step of data pre-processing, a smoothing method is performed on the subset to improve the quality of data and eliminate the impact of variety of packet generator distribution functions.
4. TONTA identifies a temporary CP in every selected subset based on a novel CPD method and divides the subset into two new temporary subsets called (Sec_1 and Sec_2).
5. The temporary CP is evaluated by a novel Curve Detection Method (CDM) as the first-step of CP verification. Curves are intermediate data points transforming two continuous predominant trends, but they are not a new trend as shown in Fig. 3. The CDM method helps to avoid selecting repetitive CPs or any data point in curves as CPs.

Table 2
Comparing the literature.

#		Application	Network	Cornerstone technique (technology)	Description
1	[33]	Traffic prediction	Cloud networks	ARIMA + Fuzzy regression	Reducing the need of historical data + using a novel sliding window
2	[34]	Network through put prediction	Multimedia networks	ARIMA+ Fractional ARIMA	Predict the throughput by FARIMA and RIMA as a long range-dependent process
3	[35]	Abnormal status prediction	N/A	ARIMA	Predict non-Gaussian traffic data to manage traffic congestion
4	[36]	Detection of degraded state anomalies	Cloud networks	Online ARIMA	Proposing an unsupervised anomaly event detection approach for service monitoring based on ARIMA
5	[37]	Network traffic prediction	Data centers	DWT + ARIMA + RNN	Predicting the network traffic to improve OoS and reduce cost
6	[38]	Data volume prediction	Wireless networks	First and second order statistics	Predicting amount of data that can be delivered through a shared band of a spectrum to guarantee the QoS statistically.
7	[39]	Network monitoring	SDN and NFV	Use *NetFlow*	Collecting network information through virtual routers of NetFlow
8	[48]	Cyber-threats detection	IP-based networks	Apache Kafka	Transforming traditional IP flow analyses to stream-based IP flow analysis
9	[49]	Anomaly detection	Performance metric systems	Adaptive learning+ robust control charts	Transforming traditional IP flow analyses to stream-based IP flow analysis
10	[50]	Congestion detection	FTP	Proposed CPD technique	Creating time periods, dividing data into subsets and putting each subset on the time periods
11	[51]	DDoS attacks detection	IP-based networks	Cumulative sum time series +	Detecting DDoS attacks based on extracting half interaction anomaly degree (HIAD)
12	[52]	DDoS attacks detection	IP-based networks	Artificial Neural Network (ANN)+	Combining traffic prediction and changing detection.

6. As the third-step of data pre-processing, an outlier detection method is performed to remove outliers of the *(*Sec_1 and Sec_2) to prepare the subsets to compare.
7. As the second-step of CP verification, two outlier-free subsets are compared to determine the importance degree of the temporary CP based on intensities of their trends. If the importance degree of the temporary CP passes a threshold (Imp_Thr), the CP turns into a permanent CP, otherwise TONTA stops working and returns to the first step, waiting for new OWD data gathered from the network.
8. If the temporary CP turns into a permanent CP, intensity of *Sec_2* is determined as the current network traffic behavior.
9. By selecting the permanent CP, *Sec_1* and all data points before that are removed from the dataset to avoid re-processing.

Fig. 2 shows the activity diagram of TONTA and how its steps identify the network traffic behavior in an online manner. To adapt TONTA with the properties of the network streams in IoT, we design it based on facing some specifications listed below:

- Although TONTA identifies the predominant trend changes, some events can cause jitters (temporary trend changes), e.g. performance of the communication protocols, and network congestion, to name a few. In addition, some events can cause short-term network traffic changes, e.g. packet bursting. These short-term trend changes can conceal predominant trends if they happen frequently as fluctuations in the time-series datasets. TONTA has been adapted to face these network properties based on data pre-processing and CP verification steps.
- IoT nodes can generate packets based on different distribution functions, e.g. Poisson, constant, normal and exponential. Each packet stream can changes its distribution function during the network lifetime. As an example, in healthcare IoT networks, emergency situations can cause changes in distribution function of a packet stream. TONTA addresses the challenge based on Step 3.

In the rest of this section, we explain all aforementioned steps of TONTA.

3.1. Collecting the network traffic characteristics

As the first step of pre-processing, TONTA tries to optimize the volume of the data that should be processed. First, a sampling method reduces the volume of OWD raw data to improve the time complexity of the algorithm. Eq. (1) is used for sampling when r is an OWD raw data, and s is the fixed sampling rate. In addition, to improve the time complexity and avoid unnecessary data processing, the *Interval_Thr* parameter is introduced. As shown in Fig. 2, TONTA gathers enough new sampled data before processing. Every *Interval_Thr* of d new sampled data points trigger TONTA. As an example, when *Interval_Thr = 20*, every 20 new sampled data points trigger TONTA; thereby if $s = 10$ and *Interval_Thr* = *20*, every 200 new packets trigger TONTA. The reasons of using the sampling method are listed below:

- Generally, TONTA as an online method, might be triggered after each new raw OWD data, but it causes high unnecessary data processing and resource consumption. The sampling method reduces the number of times that TONTA is triggered to process a time-series dataset.
- If TONTA is triggered for every few new raws of OWD data, there would be a high data redundancy to process in every continuous trigger. To avoid data redundancy, a high number of new data points should be appended to a trigger TONTA.

Big *Interval_Thr* can increase average delay of CP discovery, but it can also reduce the number of TONTA triggers. The delay is calculated based on the time of identifying a CP after occurring. On the contrary, big *Interval_Thr* may decrease the accuracy of TONTA, because in some cases, CPs will be concealed in a huge number of new data points.

$$d = \frac{\sum_{f=1}^{s} r_f}{s} \tag{1}$$

3.2. Online data analysis by dynamic-sliding window

To determine CPs in an online manner, data points needs to be analyzed gradually on-the-fly. As explained in Step 2, TONTA, in each trigger, processes a subset selected from the time-series dataset by a dynamic-running window. The subset should not be very big to consume a lot of processing resources. In addition, it should not be

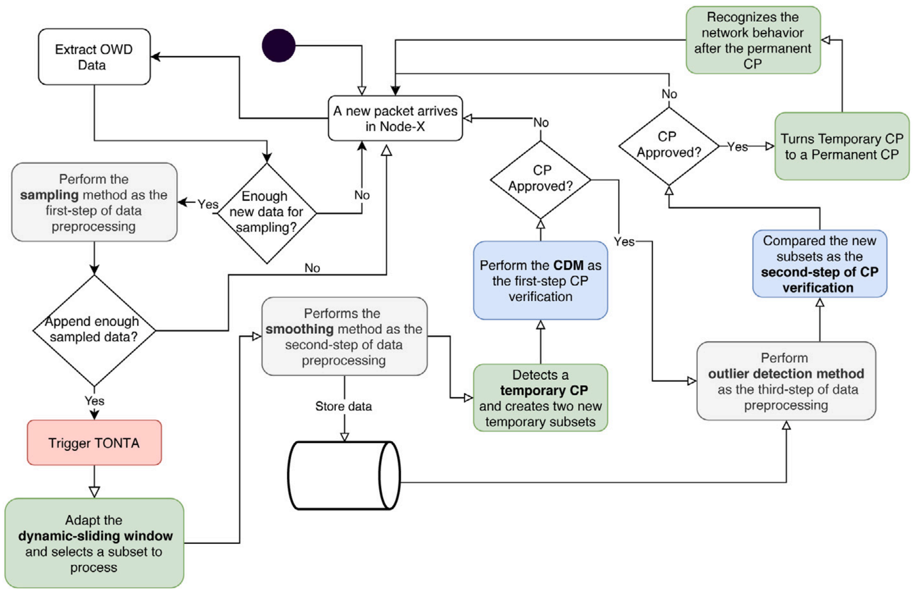
Extract OWD Data
A new packet arrives in Node-X
Recognizes the network behavior after the permanent CP
No
CP Approved?
Yes
Turns Temporary CP to a Permanent CP
No
CP Approved?
Perform the sampling method as the first-step of data preprocessing
Yes
Enough new data for sampling?
No
Yes
Compared the new subsets as the second-step of CP verification
Perform the CDM as the first-step CP verification
No
Append enough sampled data?
Performs the smoothing method as the second-step of data preprocessing
Detects a temporary CP and creates two new temporary subsets
Perform outlier detection method as the third-step of data preprocessing
Yes
Store data
Trigger TONTA
Adapt the dynamic-sliding window and selects a subset to process

Fig. 2. Activity diagram of TONTA.

very small resulting in reducing the accuracy of TONTA. To achieve low time complexity and high accuracy, the dynamic-sliding window is proposed which runs through the time-series dataset from the end and select subsets in every trigger to process. The size of the window is highly dynamic, adapted before each trigger, based on different factors like last permanent CP, data density, and $Interval_Thr$.

There are two thresholds for the dynamic-sliding window called *Max_Thr* and *Min_Thr* which show the maximum and minimum number of data points, selected by the window, respectively. *Max_Thr* limits the size of the window to improve the time complexity and resource consumption. TONTA needs a minimum number of data points to discover the network traffic pattern. *Min_Thr* is proposed to provide enough number of data points to process. Algorithm 1 shows the pseudo-code of the dynamic-sliding window to select a subset of the time-series dataset. j is a variable to improve the time complexity by removing unnecessary data points. It is used when the number of data points in the window is enough, but exceeding *Max_Thr*. j is used to keep the size of the dynamic-sliding window between *Min_Thr* and *Max_Thr* efficiently. The default value of j is 1.5. The subset selected by the window contains p data points.

Algorithm 1 Dynamic-sliding window generator

```
Input: $data\_point$ new data point extracted by SPI
Input: $Interval\_Thr$ size of sampled data to trigger TONTA
Input: $s$ sampling rate
Input: $Min\_Thr$ minimum size of the window
Input: $Max\_Thr$ maximum size of the window

  $InputDS \leftarrow \emptyset$
  $RawData \leftarrow \emptyset$
  while stream is being monitored do
     for i=1 to Interval_Thr do
        $RawData \leftarrow s$ number of new $data\_points$ extracted by SPI
        sample= Sampling($RawData$)
        $InputDS[i + size(InputDS)]$= sample
     end for
     if size($InputDS$) < Min_Thr then
        Continue
     end if
     if Size($InputDS$) > Max_Thr then
        $InputDS$=$InputDS$[Interval_Thr*j : size($InputDS$)]
     end if
     if Size($InputDS$) > Min_Thr && Size($InputDS$)<Max_Thr then
        $InputDS$=Smoothing($InputDS$)
        $CP_{Temporary}$=TONTA($InputDS$)
        if $CP_{Temporary}$ is a Permanent CP then
           $InputDS$[1 to $CP_{Temporary}$] $\leftarrow \emptyset$
        else
           Continue
        end if
     end if
  end while
```

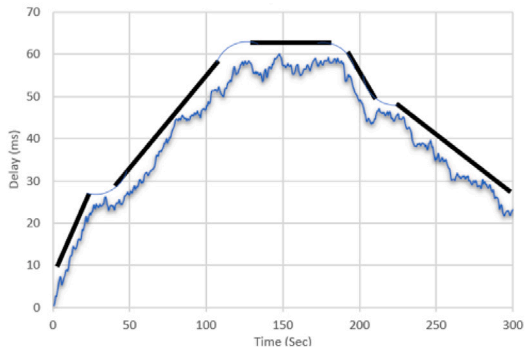

Fig. 3. Example of the Trends and Curves in a time-series dataset. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.3. Determining a temporary CP in a time-series dataset

In this step, a temporary CP in the selected subset should be selected. After appending enough new sampled data points, a smoothing method is performed on the subset to remove jitters and short-term trend changes. It also helps to transform datasets generated by different types of distribution functions to a type of dataset that can be processed by TONTA as explained in step 3. Eq. (2) is used for data smoothing based on a floating frame when the frame size is *2*m+1*. m is a parameter to change the size of the frame and d_i shows i^th data point of the subset. As an example, if $m = 1$, then the smoothing method is applied on every three sampled data points. Bigger m makes the subset smoother, but high smoothing can possibly conceal the CPs.

$$X_i = median[d_{i-m}, d_{i-m+1}, \dots, d_{i+m-1}, d_{i+m}] \qquad for\ i = 2, 3 \dots p-1 \tag{2}$$

In this step, $X_1 = d_1$ and $X_p = d_p$. To recognize a temporary CP in each trigger, as the first step, the difference of the smoothing data in the first order is used by Eq. (3).

$$T_i = X_i - X_{i-1} \qquad for\ i = 2, 3, 4 \dots p \tag{3}$$

To identify the temporary CP in the subset, we use the standard deviation method to predict existing data points and calculate the error rate of the prediction. In other words, the existing smoothed data points are predicted using the standard deviation method. Then predicted data points are compared with the smoothed data points to calculate the standard deviation errors. A novel exponential weighting method is used to give priority to the latest sampled data points. The weighting method helps to discover more important trend changes occurred recently as the current network traffic behavior. The weighting method changes the importance of each data point in the subset. Eq. (4) is used to calculate the weight for each data point.

$$\theta_{ij} = \alpha_j(1-\alpha_j)^{p-i} \qquad for\ i = 1,2,3...p \tag{4}$$

$0 < \alpha_j < 1$ is calculated by Eq. (5) to give weights to the data points of the subset. when α is approaching 1, the latest sampled data points are more important than older ones. On the contrary, if α is approaching 0, the importance of all data points tends to be equal. To identify the temporary CP, we need to give different weights from low to high to the smoothed data points to calculate the error rates. Value of α is determined by Eq. (5)

$$\alpha_j = \alpha_{j-1} + \frac{1}{\epsilon} \quad \text{for j} = 2,3,4\dots\epsilon \text{ where } \alpha_1 = \frac{1}{\epsilon} \tag{5}$$

ϵ shows the number of times that the data points are compared with different weights. Eq. (6) calculates the second order finite difference along with applying the weighting model.

$$\Delta_{ij} = \theta_{ij}(T_i - T_{i-1}) \qquad for\ i = 1,2,3...p\ \ and\ \ \text{j} = 1,2,3\dots\epsilon \tag{6}$$

Eq. (7) removes the effect of finite order difference, calculates the standard deviation error of each weighted data point, and finally sums up the error rates for each θ.

$$e_{\theta_{ij}} = \sum_{i=1}^{p}(\Delta_{ij} - T_i)^2 \qquad for\ i = 1,2,3...p\ \ and\ \ \text{j} = 1,2,3\dots\epsilon \tag{7}$$

Each $e_{\theta_{ij}}$ that shows the minimum prediction error is considered as the position of the temporary CP in the subset. As an example, if the minimum of $e_{\theta_{ij}}$ appears in $\alpha_j = 0.81$ and $p = 500$, the temporary CP is $0.81 * 500 = 405$th data point of the subset.

If $\epsilon = 100$ then TONTA compares the data points of the subset by giving 100 different weights to each data point. To reduce the time complexity of TONTA, ϵ can be increased, but the accuracy will be decreased simultaneously. Changing ϵ is based on different factors, e.g. processing power of *Node-x* as the node that is performing TONTA, sensitivity of the network traffic to discover accurate CPs, and the volume of the network traffic. It is noteworthy that if there are more than one predominant trend in the subset, the last CP is selected as the subset is weighting from the end.

3.4. Curve detection method

In computer networks, NTA methods encounter an important challenge to process time-series datasets called curves. Unlike other applications of pattern recognition techniques (e.g. signal processing), transforming two continuous network traffic behaviors happens gradually. Curves are data points between two continuous predominant trends that transform the first one to the second one, but they are not new network traffic behaviors individually. Examples of curves and trends are shown in Fig. 3. Thick (black) lines on top of the time-series dataset show different predominant trends of the dataset and narrow (blue) lines show subsets of the dataset as curves. To avoid selecting data points of curves as a predominant trend, the *Curve_Thr* variable is introduced. Avoiding the selection of CPs in curves is considered as the first-step of CP verification. As shown below, there are two conditions (C1 and C2) that should be satisfied to approve that the temporary CP is not in a curve. *Len(x)* returns the number of data points of *x* and *Pos(y)* shows the position of data point *y* in the time-series dataset. CP_{last} shows the last discovered permanent CP. The first new subset called Sec_1 is started from the first data point of the subset to the temporary CP. The second new subset called Sec_2 is started from the temporary CP to the end of the subset.

- $C1 : ((Len(Sec_1) > Curve\_Thr) \&\& (Len(Sec_2) > Curve\_Thr))$
- $C2 : (Pos(CP_{Temporary}) > (Pos(CP_{last}))$

This method is called CDM. $C1$ helps to avoid identifying curves as predominant trends. $C2$ helps to avoid selecting a CP before the last CP.

3.5. Verifying the temporary CP

After identifying the temporary CP, TONTA needs to determine the importance degree of the CP to turn it into a permanent CP or refuse it as a false alarm. It is noteworthy that the proposed method always determines a temporary CP in each trigger, but it can be false or insignificant due to the intense of the network traffic change. Therefore, Sec_1 and Sec_2 need to be compared to identify that how important the change is. As TONTA is very sensitive in terms of outliers, before comparing, Sec_1 and Sec_2 should be prepared. To address the challenge, an outlier detection method is proposed based on giving scores to data points. The assigned score is compared with a threshold to recognize and remove outliers. Eq. (8) calculates the slope of Sec_1 and Sec_2 when $y = mx+b$, where m is the slope of the desired line and b represents its intercept. Assuming that the line slope is known, one can claim that the new line equation with x is replaced with the mean of $x's$ that should result in the mean of $y's$. The score is calculated using Eq. (9) for every data point based on detrending the subset. Finally, Eq. (10) is used to determine whether each data point is an outlier or not. If the result of the Eq. (10) for a data point is 1, it is considered as an outlier and removed from the subset before comparing Sec_1 and Sec_2.

$$Slope = m = \frac{\sum_{i=1}^{x}(x_i - x)(y_i - y')}{\sum_{i=1}^{x}(x_i - x')^2} \tag{8}$$

$$Scores = |d - y| \tag{9}$$

$$f(x) = \begin{cases} 0 & \forall x | Score(x) \le Avg(Scores) + \sqrt{Var(Scores)} \\ 1 & \forall x | Score(x) > Avg(Scores) + \sqrt{Var(Scores)} \end{cases} \tag{10}$$

After removing the possible outliers, Eqs. (13), (12) and (14) are used to compare the Sec_1 and Sec_2.

$$C_{Sec_1} = \frac{1}{Len(Sec_1)} \sum_{i=2}^{Pos(CP_{temporary})} (X_i - X_{i-1}) \tag{11}$$

$$C_{Sec_2} = \frac{1}{Len(Sec_2)} \sum_{i=Pos(CP_{temporary+1})}^{p} (X_i - X_{i-1}) \tag{12}$$

$$Diff = ABS(C_{Sec_1} - C_{Sec_2}) \tag{13}$$

$$CP = \begin{cases} \text{If Diff } \ge ABS(c_{Sec_1} * \text{Imp\_Thr}) & 1 \\ \text{If Diff } < ABS(c_{Sec_1} * \text{Imp\_Thr}) & 0 \end{cases} \tag{14}$$

$Diff$ shows the intensity of the difference between trends of Sec_1 and Sec_2. In addition, C_{Sec_2} shows the behavior of the trend as the current network traffic behavior. Eq. (14) shows the final result of the TONTA method in each trigger. If its result equals 1 for a CP, the method turns the temporary CP to a permanent CP. Imp_Thr is a parameter which adapts the sensitivity of the TONTA model to turning a temporary CP into a permanent CP. As an example, if $Imp\_Thr - 0.5$, the difference between the trends of Sec_1 and $Sec2$ should be at least 50% to select the temporary CP as a permanent CP. In addition, C_{Sec_2} can have a negative or a positive value. A negative value shows the delay of the packet stream is decreasing and a positive value shows that the delay of the packet stream is increasing. When it is approaching 0, it shows that the pattern is side-way and means that the delay is stable.

After determining the temporary CP and passing the first and second verification steps, Sec_1 and all data points before the Sec_1 will be removed from the time-series dataset. In other words, if the proposed method selects a new permanent CP, the data points before the CP will not be processed again. Furthermore, after selecting a new permanent CP, if the selected subset is bigger than Max_Thr (it happens if $Sec_2 + Interval\_Thr > Max\_Thr$), then $Max\_Thr\%Interval\_Thr$ extra data points will be removed from the start of the subset. Removing the extra data means there are enough data points in the window, thereby there is no need for more data points to discover the CP. The extra data points are just removed from the window. The original data is kept to be used to determine permanent CPs shown in Fig. 2. If $Sec_2 + Interval\_Thr > Max\_Thr$, data points will be removed as extra data.

4. Evaluating the performance of TONTA

To evaluate the efficiency of TONTA, we use the Riverbed modeler (OPNET modeler) to gradually generate the time-series dataset in an ad-hoc IoT network. Two nodes are the source and destination of a packet stream, and three nodes are middle nodes to apply the effects of data forwarding queues. The forwarding rate of every middle node is 100 Pkt/Sec. The destination node is responsible for gathering OWD of every packet by SPI and sending it directly to the MATLAB engine where TONTA is implemented.

Table 3 shows the distribution functions of the packet generator at the source node for 1000 s. The number of raw data points (i.e., number of generated packets) without using the sampling method is 99 800 in 1000 s. After using sampling methods, 1914 data points are extracted gradually. The sampling rate is a constant equal to $s = 50$. The table also shows the expected network traffic behavior based on changing the distribution function, including Uptrend (UP), Downtrend (DO), Side-way (S), Very (V), Low (L), Medium (M), and High (H). As an example, we expect that after 200th second, the behavior of the network traffic changes from side-way to downtrend (DO) with a high (H) intensity, so it is (DO-H). We evaluate all public datasets to mimic all possible situations in an ad-hoc IoT network. To the best of our knowledge, no existing benchmark dataset provides the characteristics of our generated dataset listed below:

- The dataset provides different network traffic changes like abrupt changes, gradual changes, short-term changes, jitters, downtrend to uptrend transformation and vice versa, side-way to downtrend and uptrend transformations, and vice versa. The generated dataset can represent all possible transformations to evaluate the efficiency of TONTA.
- The packet generator's distribution function changes frequently to provide a dataset based on different distribution functions, e.g. normal, Poisson, constant, and exponential.

As there is no public dataset to mimic all types of transformations and distribution functions, we generate the dataset using the OPNET simulator. The x-axis of all figures of the performance evaluation section is "$Time(Second)$", whereas the y-axis denotes *"OWD (10 ms)"*. In all figures, the red labels show permanent CPs based on Eq. (14) and the red values show the exact time of every CP.

4.1. The effects of TONTA parameters

In this section, we evaluate the effects of all parameters mentioned in Table 9. As it is impossible to evaluate all possible values of the parameters, we have assigned some default values to the parameters. For each performance figure, the effects of changing the value of the parameter, which is under evaluation, is investigated. The default values are *Min_Thr = 100, Max_Thr = 300, Interval_Thr = 50, Imp_Thr = 0.5 and* $\epsilon = 100$. The reasons for assigning these values are discussed in Section 4.1.1. The following issues should be considered to evaluate the performance of TONTA:

Table 3
Distribution functions of packet generator at source node to generate the evaluated packet stream.

Start time	Distribution function	Parameter (s)	Traffic behavior
0	Exponential	0.00980392	UP-H
100	Exponential	0.00990099	S-M
200	Normal	0.01030927 0.001030927	DO-H
300	Constant	0.01	S-H
400	Poisson	0.00952380	UP-VH
500	Exponential	0.01010101	DO-L
600	Exponential	0.00970873	UP-M
700	Normal	0.00952380 0.000952380	UP-M
800	Constant	0.01	S-H
900	Poisson	0.01010101	S-L

- TONTA needs to determine CPs every 100 s, but according to the effects of curves and forwarding queues, it is practically impossible to recognize CPs in the destination node right after changing the distribution function in the data generator of the source node. After each change in the distribution function of the packet generator, it takes some time to have a considerable network traffic change in OWD of packets at the destination node.
- As an online method, it is not expected that TONTA recognizes the current network traffic behavior exactly after arising the CP as it needs some new data points after the new CP to extract the new network traffic behavior.
- As the distribution functions of the packet generator changes every 100 s, we expect the methods to identity CPs close to the time of the changes. By considering the effects of queuing in middle nodes and curves, to evaluate the efficiency of techniques, we accept alarms 30 data points before or after the time of changing the distribution function. As TONTA is an online method, it may select CPs before curves, therefore we accept CPs that are up to 30 data points before changing.

The effects of Interval_Thr. Table 4 shows the effects of changing the value of $Interval\_Thr$ from 10 to 100 each step 10. The table shows that by assigning a very small value to $Interval\_Thr$, TONTA gets stuck in short-term trend changes because it has no time to wait for new data points. In addition, having a small $Interval\_Thr$ can make a high negative impact on the performance of TONTA because TONTA needs to process the same data points several times in each triggering. A small $Interval\_Thr$ causes concealing the CPs in new data points. As shown in the table, the number of false-positive alarms is very high when $Interval\_Thr$ is small, whereas TONTA misses the CPs when $Interval\_Thr$ is big.

The effects of ϵ. ϵ is a parameter to improve accuracy and time complexity of TONTA by reducing the number of times that data points should be compared to detect a CP in a subset. Fig. 4a shows that some CPs are missed at $\epsilon = 25$. As an example, in 350th as well as 850th seconds, there are abrupt changes that TONTA cannot recognize. Valuing ϵ is related to the density of data so that ϵ cannot be very small because it causes concealing CPs and cannot be very large because it negatively affects the time complexity. Figs 4b and 4c show that increasing the value of ϵ improves the accuracy of TONTA. As shown in Table 5, increasing the value of ϵ can reduce the false negative alarms and increase the true alarms. The results show that higher ϵ improves the accuracy of TONTA, but based on the time complexity of TONTA, it also increases the time complexity simultaneously.

The effects of dynamicity of the window size. To analyze the effects of the window size, specifically Max_Thr and Min_Thr, TONTA has been performed for more than 50 times based on different parameter values. The value of Min_Thr is between 50 and 300, each step 50, and the value of Max_Thr is between 100 and 600, each step 50. We do

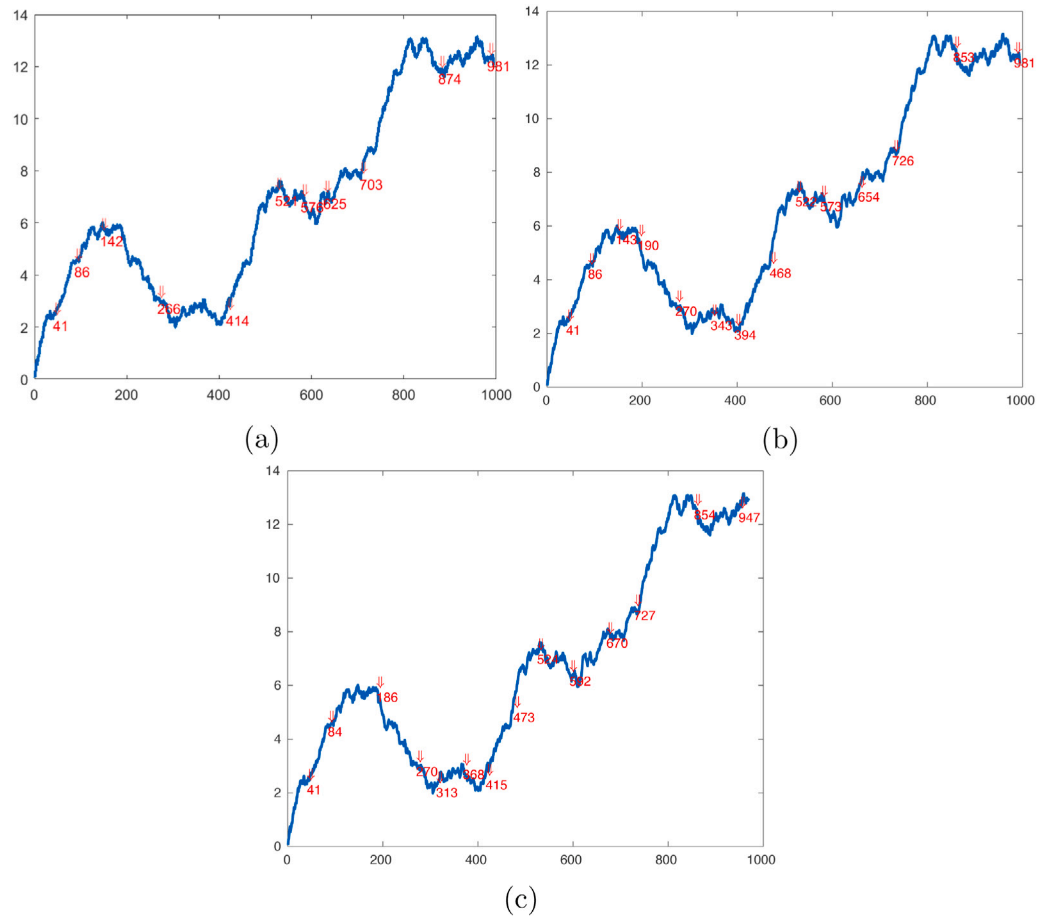
Fig. 4. The results of TONTA for default parameters values and (a) ϵ = 25; (b) ϵ = 50 and (c) ϵ = 75.

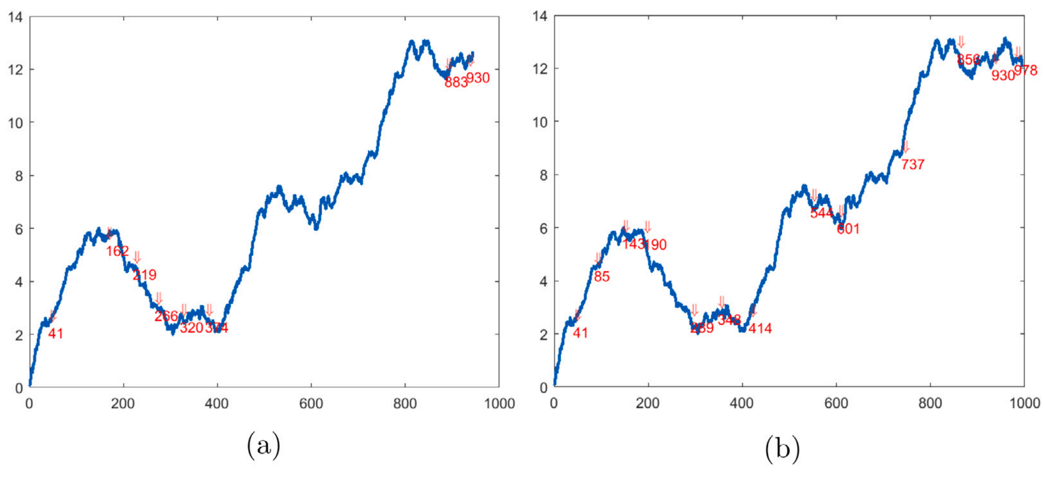
Fig. 5. The results of TONTA for default parameters values and (a) Min_Thr = 100 and Max_Thr = 150; (b) Min_Thr = 100 and Max_Thr = 600.

not consider scenarios where $Min\_Thr \geq Max\_Thr$. The results show that when these two parameters are close to each other, the number of false alarms increases. Although TONTA can recognize most of the abrupt changes in this case, for subsets that contain long curves, it is unable to recognize CPs. When Max_Thr is much bigger than Min_Thr, the results are more reliable, but it is obvious that smaller values of Max_Thr can improve the time complexity of the algorithm as shown in Section 4.3. To show the performance of the window parameters, we have selected three types of parameter values to compare the results.

Fig. 5a shows the results for small Min_Thr and Max_Thr where their values are close to each other. As shown in the figure, TONTA is unable to recognize abrupt changes and also the number of false alarms is very high. The reason is that, by assigning two close values to these parameters, the dynamic-sliding window removes older data points very fast to provide enough space in the window for new data points. Thus, TONTA has not much chance to compare new data points with previous ones. Fig. 5b shows that when Max_Thr is greater than Min_Thr, the results are highly accurate. By having a high Max_Thr, TONTA does not remove data points for a long time and compare new

Table 4
The effects of changing the $Interval\_Thr$ on the results of TONTA.

Interval_Thr	# of CPs	CPs positions	False negative alarm	False positive alarm	True alarm
10	23	81, 124, 165, 213, 260, 303, 354, 451, 500, 541, 569, 615, 662, 700, 761, 801, 845, 893, 940, 985	11	1	9
20	18	78, 121, 158, 226, 290, 334, 382, 461, 521, 571, 673, 713, 774, 819, 854, 919, 963	8	0	10
30	15	105, 189, 209, 294, 333, 409, 455, 521, 586, 641, 705, 767, 845, 904, 971	6	1	9
40	13	85, 146, 198, 305, 362, 419, 516, 595, 625, 712, 836, 922	4	1	9
50	13	41, 85, 143, 190, 289, 342, 414, 544, 601, 737, 856, 930, 978	5	3	7
60	10	138, 195, 259, 351, 412, 530, 628, 746, 813, 932	5	4	6
70	5	118, 365, 414, 718, 839	2	7	3
80	6	188, 303, 420, 584, 867, 918	1	6	4
90	5	113, 205, 302, 712, 935	1	6	4
100	5	82, 125, 173, 383, 942	2	7	3

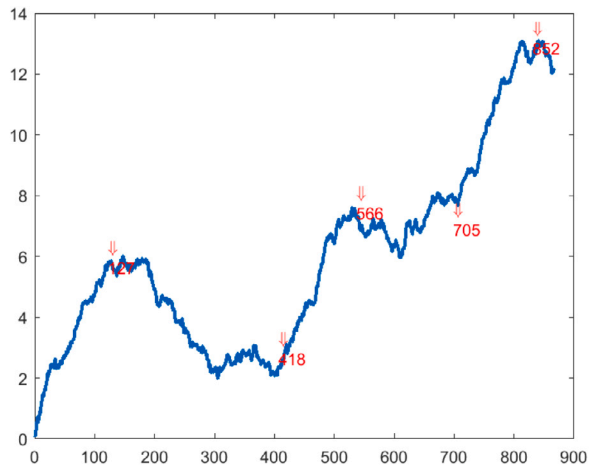

Fig. 6. The results of TONTA for default parameters values and Min_Thr = 300 and Max_Thr = 600.

Table 5
Comparing TONTA and RuLSIF.

	$\epsilon = 25$	$\epsilon = 50$	$\epsilon = 75$
False positive alarm	3	6	7
False negative alarm	3	1	1
True alarm	7	9	9

data points with old ones several times, increasing the time complexity of the algorithm. Although a high Max_Thr can improve the results, the time complexity of TONTA would be a challenge. Fig. 6 shows that, when both Min_Thr and Max_Thr are high, TONTA is unable to identify predominant trends based on Table 3. The reason of missing the CPs is that a high number of data points in the window can conceal the CPs. Therefore, it needs to have a smaller Min_Thr to trigger TONTA.

The effects of Imp_Thr. Imp_Thr is also used to change the sensitivity of TONTA in terms of identifying CPs. Smaller values of Imp_Thr helps TONTA recognize changes with a low intensity as CPs. Higher values cause considering a change point as a permanent CP if it happens because of an abrupt change. Fig. 7a shows the results of TONTA for *Imp_Thr = 0.2*. As shown, most of selected CPs connect two trends with no big difference between their intensities. The main reason that smaller *Imp_Thr* cannot identify the right CPs is that TONTA selects short-term changes and curves as predominant trends. Selecting these trends can cause concealing the real predominant trends based on CDM. Fig. 7b shows the results when $Imp\_Thr = 0.8$, in which only abrupt changes are considered as permanent CPs and gradual changes are concealed by other data points. Fig. 7c shows the results when TONTA selects changes for which the difference between intensity of the two continuous trends is 100%. As shown in the figure, only abrupt changes are selected as TONTA is unable to detect short-term and predominant trends in this case.

4.1.1. *Optimal values of parameters of TONTA*

Based on the results of evaluating different parameters, we have extracted some rules as listed below:

- Based on parameters ϵ and window size, TONTA needs to analyze a minimum number of data points in each trigger equal to Min_Thr for ϵ number of times. In this case, the value of ϵ needs to be adapted based on the value of Min_Thr to optimize the time complexity. If the value of Min_Thr is very high, ϵ should be small. On the contrary, if Min_Thr is not very high, ϵ can be a bit bigger to improve accuracy, therefore there is a trade-off between ϵ and Min_Thr.
- As shown for window size, Max_Thr has a high correlation with Min_Thr, where their values should not be very close. In addition, big values of Max_Thr can cause inefficiency in terms of the time complexity.
- The best results for $Interval\_Thr$ emerges when $Interval\_Thr = 40$ which is approximately equal to $\frac{Min\_Thr}{2}$. In the case of very big $Interval\_Thr$, important trend changes are concealed by curves. The results shown in Table 4 indicate that the best value for $Interval\_Thr$ is half of Min_Thr, improving the time complexity as well as accuracy.
- The value of Imp_Thr can be adapted based on the number of outliers removed in the second step of CP verification. If $\frac{number\ of\ outliers}{p}$ is high, then Imp_Thr should be small. On the contrary, the small number of outliers shows that the packet stream cannot have many jitters. Generally, 50% is a good value to cover all possible situations.
- $Interval\_Thr$ is related to Min_Thr, where the best results are shown for $Interval\_Thr * 2 \approx Min\_Thr$.

Based on aforementioned rules, we assign the values of *Min_Thr = 100, Max_Thr = 300, Interval_Thr = 50, Imp_Thr = 0.5 and* $\epsilon = 100$. Fig. 8 shows the results of executing TONTA based on default parameter values.

By considering that the time-series dataset is gathered gradually in an online manner, TONTA can recognize important changes without the need to wait long after each CP. The time between arising a CP and identifying it is related to the Max_Thr and $Interval\_Thr$ parameters. In TONTA, a CP should be determined after arising in a maximum Max_Thr number of new data points; otherwise, the CP will be removed. Table 6 shows details of selecting the permanent CPs based on Fig. 8. The table shows how many data points are appended to the dataset before identifying a CP.

As the final results of TONTA, the table shows the network traffic behavior between every two continuous CPs of Fig. 8 based on Eq. (13). The number of appended data points before identifying a CP is an important metric showing the delay of TONTA. As an example, between 143th and 190th seconds, there are 111 data points, but TONTA

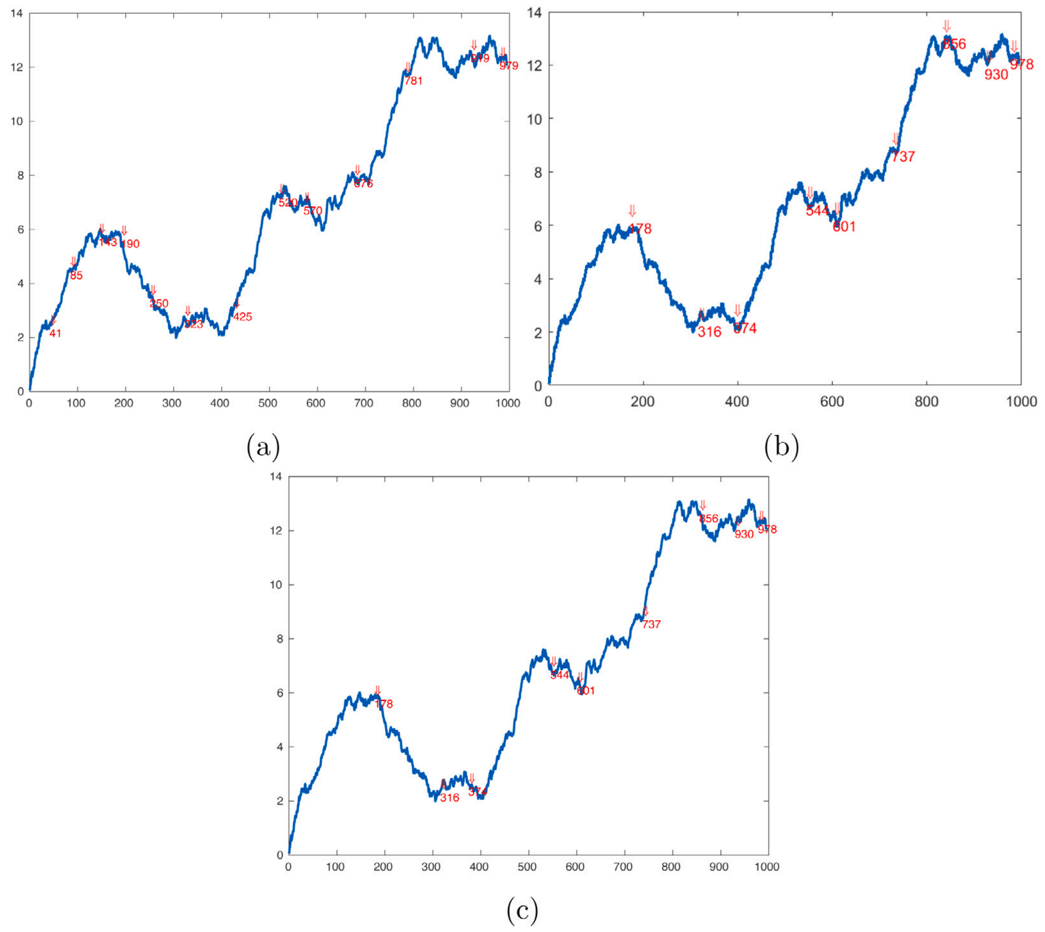

Fig. 7. The results of TONTA for default parameters values and (a) Imp_Thr = 0.2; (b) Imp_Thr = .8 and (c) Imp_Thr = 1.0.

Table 6
Network traffic behavior analyzed by TONTA and delay of TONTA to recognize CPs based on default scenario.

CP	Trends (s)	Network Traffic Behavior	Number of Data points before CP	Number of Data Points after CP	Difference
1	0–41	0.393	51	14	39%
2	41–85	0.191	117	22	−20%
3	85–143	0.147	86	13	−3.4%
4	143–190	−0.0072	92	19	−14.6%
5	190–289	−0.1405	16	33	−14%
6	289–348	0.02517	116	19	16.5%
7	348–414	−0.0405	85	18	−2.5%
8	414–544	0.20973	224	34	21%
9	544–601	−0.0422	155	31	−20%
10	601–737	0.09696	202	29	14%
11	737–858	0.12498	196	32	3%
12	858–930	−0.0517	107	26	−17.5%
13	930–1000	0.00397	77	16	5.4%

recognizes the CP at 190th second, after appending 19 new sampled data points. Network traffic behavior is also approximately 0, which means that the delay is stable.

Table 6 shows that TONTA as an online method is fast enough to recognize important changes. In most cases, TONTA identifies CPs in $\frac{1}{2} \times Interval\_Thr$ number of new data points. The average number of data points after CP is 23.5, where $Interval\_Thr = 50$. The results show that the delay of TONTA to identify CPS is related to $Interval\_Thr$. If the value of $Interval\_Thr$ is very small, TONTA cannot discover the CPs right after appending new data points and needs to wait for more data, therefore the current trigger is useless and consumes resources without any significant results. Moreover, the value of $Interval\_Thr$ should not be very high, resulting in concealing the CP among a high volume of new data points.

4.2. Benchmarking

We compare TONTA with RuLSIF [56] as a benchmark TCD algorithm to process time-series datasets. RuLSIF has been widely used for TCD in different applications, e.g. computer networks [57], healthcare [58], and daily activity segmentation [59], to name a few. In addition, several state-of-the-art TCD techniques are compared with RuLSIF to evaluate their performance [23,57,60–63]. It is noteworthy that RuLSIF, as an offline method, needs the whole dataset at once to process. Although comparing an offline and an online method is not the best option, it can express the robustness of TONTA. In addition, RuLSIF has been selected to be compared with TONTA because of their similarities, e.g. both methods use the running window and try to reduce overlapped data points.

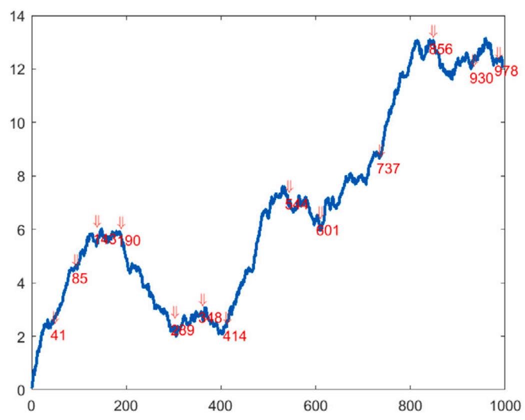

Fig. 8. The results of the TONTA for default parameter values.

Table 7
The results of RuLSIF based on Fig. 9.

CP	Trends (s)	Network traffic behavior	Difference
1	0–45	0.382	38%
2	45–88	0.201	−18%
3	88–90	0.41	Absolutely false
4	90–202	0.0523	−35%
5	202–250	−0.173	−22.5%
6	250–251	−0.2	Absolutely false
7	251–297	−0.148	5%
8	297–402	0.025	17%
9	402–447	0.353	13%
10	447–497	0.382	2.7%
11	497–499	−0.35	Absolutely false
12	499–538	0.017	36%
13	538–602	−0.05	−6%
14	602–603	0.26	Absolutely false
15	603–652	0.04	−22%
16	652–697	0.07	%3
17	697–744	0.09	%2
18	744–745	−0.4	Absolutely false
19	745–808	0.323	36%
20	808–860	0.21	−11%
21	860–902	−0.06	−28%
22	902–903	0.42	Absolutely false

RuLSIF is used to assess the ratio of relative density among subsequences of a time-series dataset. It uses divergence of data points to measure the relative density. Three parameters are important to adapt RuLSIF, including n: the number of overlapped windows, k: the size of each overlapped window, and a: the relationship among data points. If the data points in a time-series dataset are highly related to each other, a higher a is needed. Using a supervised learning method, we have extracted values as $n = 100$, $k = 10$ and $a = 0.1$ based on our generated time-series dataset.

Fig. 9 shows the results of the RuLSIF method. Red peaks show the scores of RuLSIF for data points as RuLSIF assigns a score to each data point. 0.1 is considered as the threshold to recognize CPs, therefore most of the peaks are considered as change points. Table 7 also shows the results of RuLSIF in details. We use the same method for RuLSIF as we used in TONTA to calculate network traffic behavior between two continuous CPs.

As shown in Table 7, there are some absolutely false alarms showing that RuLSIF detects a CP very close to another CP. Comparing Tables 3, 6 and 4 shows that TONTA has a high accuracy compared to RuLSIF. Although it is difficult to calculate the accuracy of both methods because of the delays of queues in middle nodes, comparing the tables shows that TONTA detects better CPs compared to RuLSIF based on expected network traffic behaviors mentioned in Table 3. As an example, it is expected that both TONTA and RuLSIF detect network traffic behavior between 600th and 700th as an uptrend with high

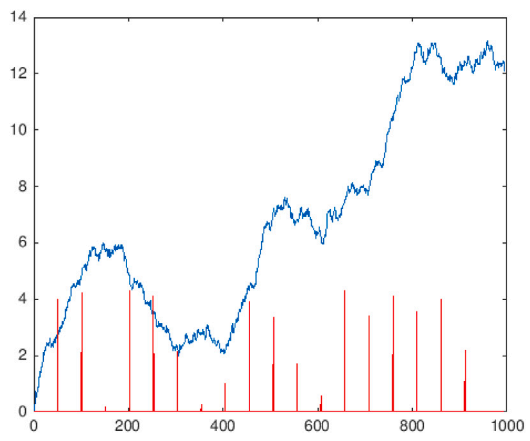

Fig. 9. The results of RuLSIF ($n = 100$, $k = 10$ and $a = 0.1$). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 8
Comparing TONTA and RuLSIF.

	TONTA	RuLSIF	Efficiency of TONTA compared to RuLSIF
False positive alarm	5	13	61%
False negative alarm	3	1	−20%
True alarm	7	9	−20%

intensity. TONTA detects two CPs in 601th and 737th seconds, and the network traffic behavior between them is 0.09696, showing that the delay is increasing. As the maximum network traffic behavior is 0.5, the result shows 20% increase in the delay of every packet which is high. RuLSIF detects four CPs in 602th, 603th, 652th, and 697th seconds and behaviors of the network traffic between each two continuous CPs are 0.26, 0.04, and 0.07, respectively. Even if we consider the average of values, it is 0.123, but the number of false alarms is very high as we change the rate of the packet generator one time, but we receive four alarms from the CPD model.

As the number of false alarms in RuLSIF is very high, they conceal the important network traffic changes and make it hard to trust the results. In addition, divergence of each two continuous changes that TONTA selects is higher than RuLSIF based on comparing the *Difference* field in Tables 6 and 7. The comparison shows that TONTA selects more important network traffic behaviors as there are some small values in the results of RuLSIF, e.g. 2% and 5%. Comparing TONTA and RuLSIF shows that the accuracy of TONTA is higher than RuLSIF based on the positions of CPs, which are recognized in a common time-series dataset. We consider three types of alarms to compare the efficiency of TONTA and RuLSIF, as shown in Table 8. As the application of TONTA is NTA, false alarms can highly affect the performance of the network.

4.3. Time complexity of TONTA

Based on Section 4.1, we evaluate the effect of the parameters on the accuracy of TONTA. In addition, we calculate the time complexity of TONTA to show how the proposed method performs on weak machines. The time complexity of the sampling method is *O(1)* and the smoothing method is *O(Max_Thr)*. In terms of detecting the temporary CP, the time complexity is equal to *O(Max_Thr*(ϵ))*. Time complexity of outlier detection is also *O(Max_Thr)*, therefore the whole time complexity of TONTA is equal to *O((2*Max_Thr)+Max_Thr*(ϵ))*. *2* is removed from the time complexity as a constant. Thus, the final time complexity would be *O(Max_Thr+(Max_Thr*ϵ))*.

Table 9 shows different parameters of TONTA and their effects on the time complexity of the proposed method and its accuracy in terms of TCD based on Section 4.1.

Table 9
Parameters of TONTA and their effects on time complexity and accuracy.

Parameter	Accuracy	Time complexity
ϵ	High	High
Max_Thr	Very low	High
Min_Thr	High	Low
Imp_Thr	High	Very low
Interval_Thr	High	Low

5. Conclusions and future work

NTMA is not a mature field in ad-hoc IoT networks as today's network resources are not capable of executing complicated NTMA techniques. Although different NTMA techniques have been introduced based on machine learning solutions, they suffer from high time-complexity of training and the need for labeled data. To address the aforementioned challenges, we focus on using statistical NTA techniques. We proposed TONTA to enable IoT nodes to recognize abrupt or gradual network traffic changes based on TCD techniques. TONTA proposes a lightweight method integrated with a dynamic-sliding window to reduce time complexity and resource consumption, along with improving the accuracy of NTA. Unlike most of the trend change detection methods, TONTA does not need to process a large amount of data at once to detect CPs, while it can detect CPs in an online manner based on time-series datasets. It uses a dynamic-running window to select data for processing, which makes it compatible with weak nodes of IoT. The results of TONTA can be used to improve QoS and security of IoT networks. TONTA can be used hop by hop in a route as a distributed method to recognize the location of events that cause network behavior changes, which is part of our future work. Using TONTA as a distributed method to discover the reasons for the changes is also considered as our future work. In addition, TONTA can be improved in terms of security to handle false data injection attacks or manipulative attacks.

CRediT authorship contribution statement

Amin Shahraki: Conceptualization, Methodology, Writing - original draft, Writing - review & editing, Data curation, Resources, Formal analysis, Software, Investigation, Visualization. **Amir Taherkordi:** Writing - review & editing, Validation, Supervision. **Øystein Haugen:** Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] CR Srinivasan, et al., A review on the different types of internet of things (iot), J. Adv. Res. Dyn. Control Syst. 11 (1) (2019) 154–158.
[2] Amin Shahraki, Øystein Haugen, Social ethics in internet of things: An outline and review, in: 2018 IEEE Industrial Cyber-Physical Systems (ICPS), IEEE, 2018, pp. 509–516.
[3] Jasenka Dizdarević, Francisco Carpio, Admela Jukan, Xavi Masip-Bruin, A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration, ACM Comput. Surv. 51 (6) (2019) 1–29.
[4] O. Bello, et al., Intelligent device-to-device communication in the internet of things, IEEE Syst. J. 10 (3) (2014) 1172–1182.
[5] Amin Shahraki, Mahmoud Abbasi, Md Piran, Mingzhe Chen, Shuguang Cui, et al., A comprehensive survey on 6g networks: Applications, core services, enabling technologies, and future challenges, 2021, arXiv preprint arXiv:2101.12475.
[6] Amin Shahraki, Amir Taherkordi, Øystein Haugen, Frank Eliassen, Clustering objectives in wireless sensor networks: A survey and research direction analysis, Comput. Netw. 180 (2020) 107376.
[7] Amin Shahraki, Amir Taherkordi, Øystein Haugen, Frank Eliassen, A survey and future directions on clustering: From wsns to iot and modern networking paradigms, IEEE Trans. Netw. Serv. Manag. (2020).
[8] A. Dhumane, et al., Routing issues in internet of things: a survey, in: Proceedings of the International Multiconference of Engineers and Computer Scientists, Vol. 1, 2016, pp. 16–18.
[9] A. D'Alconzo, et al., A survey on big data for network traffic monitoring and analysis, IEEE Trans. Netw. Serv. Manag. 16 (3) (2019) 800–813.
[10] Tiwari Nitin, Sr Singh, Pg Singh, Intrusion detection and prevention system (idps) technology-network behavior analysis system (nbas), Int. Sci. Congr. Assoc. 1 (1) (2012) 51–56.
[11] Yuri Breitbart, Minos Garofalakis, Ben Jai, Cliff Martin, Rajeev Rastogi, Avi Silberschatz, Topology discovery in heterogeneous ip networks: the netinventory system, IEEE/ACM Trans. Netw. 12 (3) (2004) 401–414.
[12] Mahmoud Abbasi, Amin Shahraki, Amir Taherkordi, Deep learning for network traffic monitoring and analysis (ntma): A survey, Comput. Commun. (2021).
[13] Amin Shahraki, Mahmoud Abbasi, Ø ystein Haugen, Boosting algorithms for network intrusion detection: A comparative evaluation of real adaboost, gentle adaboost and modest adaboost, Eng. Appl. Artif. Intell. 94 (2020) 103770.
[14] L. Bilge, et al., Exposure: Finding malicious domains using passive dns analysis., in: Ndss, 2011, pp. 1–17.
[15] F. Jemili, et al., A framework for an adaptive intrusion detection system using bayesian network, in: 2007 IEEE Intelligence and Security Informatics, IEEE, 2007, pp. 66–70.
[16] James V Hansen, Paul Benjamin Lowry, Rayman D Meservy, Daniel M McDonald, Genetic programming for prevention of cyberterrorism through dynamic and evolving intrusion detection, Decis. Support Syst. 43 (4) (2007) 1362–1374.
[17] Mohammad Abu Alsheikh, et al., Mobile big data analytics using deep learning and apache spark, IEEE Netw. 30 (3) (2016) 22–29.
[18] Chaoyun Zhang, Paul Patras, Hamed Haddadi, Deep learning in mobile and wireless networking: A survey, IEEE Commun. Surv. Tutor. 21 (3) (2019) 2224–2287.
[19] D. Zhou, et al., A survey on network data collection, J. Netw. Comput. Appl. 116 (2018) 9–23.
[20] M. Conti, et al., The dark side (-channel) of mobile devices: A survey on network traffic analysis, IEEE Commun. Surv. Tutor. 20 (4) (2018) 2658–2713.
[21] Jessica Lin, Sheri Williamson, Kirk Borne, David DeBarr, Pattern recognition in time series, Adv. Mach. Learn. Data Min. Astron. 1 (617–645) (2012) 3.
[22] S. Jamali, et al., Detecting changes in vegetation trends using time series segmentation, Remote Sens. Environ. 156 (2015) 182–195.
[23] Samaneh Aminikhanghahi, Diane J Cook, A survey of methods for time series change point detection, Knowl. Inf. Syst. 51 (2) (2017) 339–367.
[24] Bruce Ratner, Statistical and Machine-Learning Data Mining:: Techniques for Better Predictive Modeling and Analysis of Big Data, CRC Press, 2017.
[25] Amin Shahraki, Hamed Taherzadeh, Ø ystein Haugen, Last significant trend change detection method for offline poisson distribution datasets, in: 2017 International Symposium on Networks, Computers and Communications, ISNCC 2017, 2017, http://dx.doi.org/10.1109/ISNCC.2017.8071994.
[26] Amin Shahraki, Ø ystein Haugen, An outlier detection method to improve gathered datasets for network behavior analysis in iot, J. Commun. 14 (6) (2019) 455–462.
[27] Pedro Casas, Alessandro D'Alconzo, Tanja Zseby, Marco Mellia, Big-DAMA: big data analytics for network traffic monitoring and analysis, in: Proceedings of the 2016 Workshop on Fostering Latin-American Research in Data Communication Networks, 2016, pp. 1–3.
[28] Y. Meidan, et al., Profiliot: a machine learning approach for iot device identification based on network traffic analysis, in: Proceedings of the Symposium on Applied Computing, ACM, 2017, pp. 506–509.
[29] Arijit Ukil, Soma Bandyoapdhyay, Chetanya Puri, Arpan Pal, Iot healthcare analytics: The importance of anomaly detection, in: 2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA), IEEE, 2016, pp. 994–997.
[30] M. Iqbal, et al., Efficient prediction of network traffic for real-time applications, J. Comput. Netw. Commun. 2019 (2019).
[31] Daniel R Jeske, Nathaniel T Stevens, Alexander G Tartakovsky, James D Wilson, Statistical methods for network surveillance, Appl. Stoch. Models Bus. Ind. 34 (4) (2018) 425–445.
[32] Marjan Kuchaki Rafsanjani, Atieh Rezaei, Amin Shahraki, Arsham Borumand Saeid, Qarima: A new approach to prediction in queue theory, Appl. Math. Comput. 244 (2014) 514–525.
[33] H. Mehdi, et al., Cloud traffic prediction based on fuzzy arima model with low dependence on historical data, Trans. Emerg. Telecommun. Technol. (2019) e3731.
[34] Arkadiusz Biernacki, Improving quality of adaptive video by traffic prediction with (f) arima models, J. Commun. Netw. 19 (5) (2017) 521–530.
[35] Taghreed Alghamdi, Khalid Elgazzar, Magdi Bayoumi, Taysseer Sharaf, Sumit Shah, Forecasting traffic congestion using arima modeling, in: 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), IEEE, 2019, pp. 1227–1232.

[36] F. Schmidt, et al., Unsupervised anomaly event detection for cloud monitoring using online arima, in: 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion, IEEE, 2018, pp. 71–76.
[37] Rishabh Madan, Partha SarathiMangipudi, Predicting computer network traffic: a time series forecasting approach using dwt, arima and rnn, in: 2018 Eleventh International Conference on Contemporary Computing (IC3), IEEE, 2018, pp. 1–5.
[38] X. Li, et al., Statistical qos provisioning over uncertain shared spectrums in cognitive iot networks: A distributionally robust data-driven approach, IEEE Trans. Veh. Technol. 68 (12) (2019) 12286–12300.
[39] Ch. Yang, et al., Implementation of a real-time network traffic monitoring service with network functions virtualization, Future Gener. Comput. Syst. 93 (2019) 687–701.
[40] Ismail Butun, Burak Kantarci, Melike Erol-Kantarci, Anomaly detection and privacy preservation in cloud-centric internet of things, in: Communication Workshop (ICCW), 2015 IEEE International Conference on, IEEE, 2015, pp. 2610–2615.
[41] Douglas H Summerville, Kenneth M Zach, Yu Chen, Ultra-lightweight deep packet anomaly detection for internet of things devices, in: 2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC), IEEE, 2015, pp. 1–8.
[42] D. Stiawan, et al., Anomaly detection and monitoring in internet of things communication, in: Information Technology and Electrical Engineering, 8th International Conference on, IEEE, 2016, pp. 1–4.
[43] M. Behniafar, et al., A survey of anomaly detection approaches in internet of things., ISeCure 10 (2) (2018).
[44] Mahmoud Abbasi, Amin Shahraki, Md. Jalil Piran, Amir Taherkordi, Deep reinforcement learning for qos provisioning at the mac layer: a survey, Eng. Appl. Artif. Intell. 102 (2021) 104234, http://dx.doi.org/10.1016/j.engappai.2021.104234, https://www.sciencedirect.com/science/article/pii/S0952197621000816.
[45] P. Bull, et al., Flow based security for iot devices using an sdn gateway, in: 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), IEEE, 2016, pp. 157–163.
[46] Keiichi Yasumoto, Hirozumi Yamaguchi, Hiroshi Shigeno, Survey of real-time processing technologies of iot data streams, J. Inf. Process. 24 (2) (2016) 195–202.
[47] Robert Maidstone, Toby Hocking, Guillem Rigaill, Paul Fearnhead, On optimal multiple changepoint algorithms for large data, Statist. Comput. 27 (2) (2017) 519–533.
[48] T. Jirsik, et al., Toward stream-based ip flow analysis, IEEE Commun. Mag. 55 (7) (2017) 70–76.
[49] Olumuyiwa Ibidunmoye, Ali-Reza Rezaie, Erik Elmroth, Adaptive anomaly detection in performance metric streams, IEEE Trans. Netw. Serv. Manag. 15 (1) (2017) 217–231.
[50] Cecilia Dao, Xinyu Liu, Alex Sim, Craig Tull, Kesheng Wu, Modeling data transfers: Change point and anomaly detection, in: 38th International Conference on Distributed Computing Systems, IEEE, 2018.
[51] J. Cheng, et al., A change-point ddos attack detection method based on half interaction anomaly degree, Int. J. Autonom. Adapt. Commun. Syst. 10 (1) (2017) 38–54.
[52] Mouhammd Alkasassbeh, A novel hybrid method for network anomaly detection based on traffic prediction and change point detection, 2018, arXiv preprint arXiv:1801.05309.
[53] P. Barford, et al., Method and apparatus for network anomaly detection, 2017, Google Patents US Patent 9, 680, 693.
[54] A. Dhumane, et al., Routing issues in internet of things: a survey, in: Proceedings of the International Multiconference of Engineers and Computer Scientists, Vol. 1, 2016, pp. 16–18.
[55] Rasa Bruzgiene, Lina Narbutaite, Tomas Adomkus, Manet network in internet of things system, in: Ad Hoc Networks, InTech, 2017.
[56] S. Liu, et al., Change-point detection in time-series data by relative density-ratio estimation, Neural Netw. 43 (2013) 72–83.
[57] Yasmin Fathy, Payam Barnaghi, Rahim Tafazolli, An online adaptive algorithm for change detection in streaming sensory data, IEEE Syst. J. 13 (3) (2018) 2688–2699.
[58] Gina Sprint, Diane J Cook, Roschelle Fritz, Maureen Schmitter-Edgecombe, Using smart homes to detect and analyze health events, Computer 49 (11) (2016) 29–37.
[59] Samaneh Aminikhanghahi, Diane J Cook, Using change point detection to automate daily activity segmentation, in: 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), IEEE, 2017, pp. 262–267.
[60] M. Zameni, et al., Unsupervised online change point detection in high-dimensional time series, Knowl. Inf. Syst. 62 (2) (2020) 719–750.
[61] Xiaochen Zhang, Santiago Grijalva, A data-driven approach for detection and estimation of residential pv installations, IEEE Trans. Smart Grid 7 (5) (2016) 2477–2485.
[62] Samaneh Aminikhanghahi, Tinghui Wang, Diane J Cook, Real-time change point detection with application to smart home time series data, IEEE Trans. Knowl. Data Eng. 31 (5) (2018) 1010–1023.
[63] M. Zameni, et al., Change point detection for streaming high-dimensional time series, in: International Conference on Database Systems for Advanced Applications, Springer, 2019, pp. 515–519.

Amin Shahraki was born in Mashhad, Iran, 1988. He received his bachelor's degree in computer software engineering and master's degree in computer networks in 2009 and 2012 respectively. He received his Ph.D. degree in Computer networks from University of Oslo, Norway in 2020. During his Ph.D., he was also working as a research assistant at Østfold University College and working on Smart Buildings and welfare project. He has been a visiting researcher at University of Melbourne, CLOUDS Lab from Aug. 2019 to Feb. 2020 under supervision of Professor Rajkumar Buyya. He is a member of National Elite Foundation in Iran and IEEE since 2011. His current research interests are Internet of Things, Cellular network, Cognitive Communications and Networking, Network Behavior Analysis, Time Series Analysis, Clustering and QoS and self-healing networks.

Amir Taherkordi is an Associate Professor in the Networks and Distributed Systems (ND) group at the Department of Informatics, University of Oslo. Amir received his Ph.D. degree from the Informatics Department, UiO and his Ph.D. thesis was titled "Programming Wireless Sensor Networks: From Static to Adaptive Models". He has experience from several RCN and EU projects and serves as a reviewer for many high-ranking conferences in the area of distributed systems. Amir's research interests lie broadly in the area of distributed computing, software engineering, adaptation middleware, networked embedded systems (WSNs, IoT, and CPS), and Internet/Web engineering. He has recently received a prestigious grant from the Norwegian Research Council under the scheme of Young Research Talents to work on a novel IoT service computing model for future large-scale and dynamic Fog-Cloud systems.

Øystein Haugen is a professor at Østfold University College. His research focuses on cyber–physical systems, variability modeling and software product lines, object oriented languages and methods, software engineering, and practical formal methods.