IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 12, 2020, accepted November 26, 2020, date of publication December 1, 2020,
date of current version December 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3041641

Distributed Network Intrusion Detection System in Satellite-Terrestrial Integrated Networks Using Federated Learning

KUN LI, HUACHUN ZHOU, ZHE TU, WEILIN WANG, AND HONGKE ZHANG, (Senior Member, IEEE)
School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China

Corresponding author: Huachun Zhou (hchzhou@bjtu.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFA0701604, in part by NSFC under Grant 61802014 and Grant U1530118, and in part by the National High Technology of China (863 Program) under Grant 2015AA015702.

ABSTRACT The existing satellite-terrestrial integrated networks (STINs) suffer from security and privacy concerns due to the limited resources, poor attack resistance and high privacy requirements of satellite networks. Network Intrusion Detection System (NIDS) is intended to provide a high level of protection for modern network environments, but how to implement distributed NIDS on STINs has not been widely discussed. At the same time, satellite networks have always lacked real and effective security data sets as references. To solve these problems, we propose a distributed NIDS using Federal Learning (FL) in STIN to properly allocate resources in each domain to analyze and block malicious traffic, especially distributed denial-of-service (DDoS) attacks. Specifically, we first design a typical STIN topology, on the basis of which we collect and design security data sets adapted to satellite and terrestrial networks in STIN, respectively. To address the problem of poor attack resistance of satellite networks, we propose a satellite network topology optimization algorithm to reduce the difficulty in tracing malicious packets due to frequent link switching. In order to solve the problem of limited resources and high privacy requirements of satellite networks, we propose an algorithm for FL adaptation to STIN, and build a distributed NIDS using FL in STIN. Finally, we deploy the designed distributed NIDS in a prototype system and evaluate our proposed distributed NIDS with a large number of simulations of randomly generated malicious traffic. Related results demonstrate that the performance of our approach is better than traditional deep learning and intrusion detection methods in terms of malicious traffic recognition rate, packet loss rate, and CPU utilization.

INDEX TERMS Satellite-terrestrial integrated network, distributed NIDS, security data set, federated learning.

I. INTRODUCTION

As an important supplement to the wireless network, satellite-terrestrial integrated network (STIN) offers large-capacity information transmission service to space access network and terrestrial network in a recent decade [1]. However, with the development of satellite networks and billions of devices are being connected to the continuously growing networks, security and privacy appear to be a major concern in the STIN. Unfortunately, there exists a huge gap between satellite networks and terrestrial networks in computation power, bandwidth, and other resources. Besides, the hardware is difficult to be upgraded after satellites launched [2]. To prevent attacks, one method is to use keys to encrypt/decrypt and authenticate data during transmission [3], [4]. Once attacked, the resources of the satellite network will be quickly exhausted and difficult to repair, which will make it difficult to transmit the keys. Even if a firewall system is deployed in the satellite node, it is not able to detect modern attack environments and analyze network packets in depth. Because of

The associate editor coordinating the review of this manuscript and approving it for publication was Yiming Tang.

these reasons, Network Intrusion Detection Systems (NIDS) are designed to achieve high protection for the modern network environment [5].

A NIDS is designed for detecting detrimental intrusions. Generally, a NIDS is used for identifying the traffic in the network, especially for distinguishing normal and malicious traffic and hence is beneficial in eliminating malicious traffic [6]. As for the NIDS in the STIN, considering the limited computing power of each satellite node and the high security and privacy requirements in the transmission process, resource utilization efficiency should be considered when designing the detection method and system architecture. Thanks to the development of artificial intelligence, compute nodes do not need high-computing hardware configuration to effectively detect known and unknown intrusion traffic based on the database in a short time.

There have been many works proposed aiming at the design of NIDS typically invoking machine learning algorithms for identifying traffic [7]–[10]. The effectiveness of NIDS is evaluated based on their performance to identify attacks which requires a comprehensive data set that contains normal and abnormal behaviors, such as NSLKDD [11] and UNSW-NB15 [12]. However, these data sets and NIDSs are generally used for terrestrial networks and are difficult to apply to satellite communications because of the following characteristics of STIN.

- Limited resource on satellites. The energy on satellites is limited and the available resources of satellites are much lower than that of the terrestrial calculator. Meanwhile, the information transmitted through the satellite network has a higher degree of privacy. Therefore, satellite communications have higher security needs than terrestrial networks. NIDS for STINs should have lower computational complexity and higher recognition accuracy.
- Satellite networks and terrestrial networks have different tolerances for various types of attacks. Because terrestrial network nodes have the characteristics of high computing performance and high openness, attacks such as Backdoor and Botnets will bring huge trouble to the terrestrial network. However, satellite nodes with limited resources are afraid of malicious distributed denial-of-service (DDoS) attacks such as Synchronize Sequence Numbers (Syn) flooding. Careful consideration should be given to the structure of security data sets in the different domains of the STIN.
- High privacy needs. The priority of data transmitted by satellite networks and terrestrial networks is different. The content transmitted by satellite networks is generally more private, so data from two heterogeneous networks cannot be integrated. It is difficult to construct a comprehensive data set under the STIN because the data privacy of each domain needs to be guaranteed.

To solve the dilemma of data islands in heterogeneous networks, bottlenecks have arisen by traditional methods. There have been some researches that focused on Some research focuses on a feasible solution that meets privacy protection and data security, called Federated Learning (FL) [13]–[15]. Indeed, the idea of FL can solve the problem of data islands. However, the satellite network field lacks a real and effective security data set, which is insufficient to support the application of FL in STINs.

Therefore, in this paper, we are motivated to focus on distributed NIDS in STIN by deploying FL method. In response to the different tolerances of satellites and terrestrial networks to attacks, we have designed security data sets that conform to the characteristics of each domain. The data sets are collected in a large and real Linux-based prototype, which contains nearly 40 nodes and more than 10 different types of simulated attacks. We adopt DTN [16] in the satellite networks and use Tcpdump [17] to capture network traffic in the form of packets, and utilize the Argus [18] and CICFlowMeter [19] tools to create reliable features from the pcap files. Then, to combine the horizontal FL method with a NIDS in a STIN, we propose a FL adapted STIN algorithm to implement data exchange and model sharing in heterogeneous networks.

In addition, we also propose a simple satellite network topology optimization algorithm and implement it in a prototype based on Openstack. We aim to construct a topology without link switching under the limitation of satellite antenna, and hope it can stably adapt to FL-based NIDS. Our emulation results show that distributed NIDS using FL have a better performance in terms of malicious traffic recognition rate, packet loss rate, and CPU utilization than traditional deep learning methods and NIDSs.

To sum up, our main technical contributions in this work are as follows.

- We design security data sets suitable for the STIN in the prototype, borrowing the Argus and CICFlowMeter to extract highly relevant features from the pcap file.
- We propose a FL adapted STIN algorithm to combine the horizontal FL method with a NIDS in STINs. Then, we propose a satellite network topology optimization algorithm and implement it with the FL-based NIDS in a large and real Linux-based prototype, which contains nearly 40 nodes and many different types of emulated attacks.
- We provide evaluation analysis to validate the functionality of the prototype and the feasibility of the distributed NIDS using FL. The results of the emulations show that our proposed distributed NIDS can achieve higher accuracy of malicious traffic identification and lower CPU utilization than traditional NIDSs.

The rest of the paper is organized as follows. Section II reviews the related work. Section III describes how to collect and create data sets in a STIN in our prototype. Section IV proposes a FL adapted STIN algorithm to combine the horizontal FL method with the NIDS. Then, a satellite network topology optimization algorithm is proposed to generate the satellite network topology combined with NIDS and deployed in the prototype. In section V, we deploy the

distributed NIDS using FL in our Openstack-based prototype, and analyze the experimental results according to the evaluation criteria. Finally, section VI sums up the paper.

II. RELATED WORK

In this paper, we focus on designing security data sets and deploying the distributed NIDS using FL in STINs. In this section, we first review some recent related work on security data sets and FL. Then, we reviewed the latest development of NIDS applied to DDoS detection in STINs.

A. SECURITY DATA SETS

Training, testing, and evaluation of IDS with real network traffic is a significant challenge, so most IDS evaluation is based on intrusion data sets [20]. KDD'99 [21] is used in The Third International Knowledge Discovery and Data Mining Tools contest. The data sets include a total of 24 training attack types, with an additional 14 types of experimental data only. NSLKDD [11] is a data set suggested to solve some of the intrinsic problems of the KDD'99 data set, but it does not represent the modern low footprint attack scenarios. Because KDD'99 and NSLKDD realized a limited number of attacks and information of outdated packets, UNSW-NB15 [12] is created by establishing the synthetic environment at the UNSW cybersecurity lab. UNSW-NB15 represents nine major families of attacks by utilizing the IXIA PerfectStorm tool. 49 features have been developed using Argus, Bro IDS tools, and twelve algorithms that cover characteristics of network packets.

These main and commonly used data sets are focused on the research of terrestrial network security, but satellite networks cannot effectively use these data sets due to different transmission protocols and traffic environments. In this article, we research and design a STIN security data set based on our prototype.

B. FEDERATED LEARNING

In the traditional method, data collected by terminal devices is centrally uploaded and processed in a cloud-based data center. With the divergence of data sources today, MapReduce programming model [22] and distributed machine learning (DML) [23] have naturally been proposed as solutions. They mainly use the computing and storage functions of terminal devices and edge servers to train distributed data in parallel, which enables large-scale data processing across multiple fields. However, computation offloading and data processing at edge servers still involve the transmission of potentially sensitive data, especially security data used for DDoS attack detection. This can discourage privacy-sensitive consumers from taking part in model training, or even violate increasingly stringent privacy laws [24].

The concept of FL is proposed by Google recently [14]. Their main idea is to build machine learning models based on data sets that are distributed across multiple devices while preventing data leakage. This framework applies to a data-partition framework where each partition corresponds to a subset of data samples collected from one or more users. Instead of sending the raw data over for processing, participating devices only send the updated model parameters for aggregation. Compared with MapReduce and DML, FL can efficiently use network bandwidth while improving user privacy [25]. Therefore, we combine the prototype-generated STIN security data set with FL to meet the new requirements for security and privacy of heterogeneous networks.

C. NETWORK INTRUSION DETECTION SYSTEM

The rapid progress of STIN has also brought huge security risks, prompting the development of security technology, especially NIDS. Attacks as DDoS is one of the most popular intrusions in STINs, which can lead to a decline in service quality or denial of service [26]. Many studies have used different machine learning techniques to establish effective NIDSs for DDoS attack detection in satellite networks or terrestrial networks [27]–[30]. For DDoS attacks in the satellite network, Di *et al.* [27] proposed an idea of defense architectures combining with distributed multipoint detection, near-source defense, collaborative management and integrality of protection. For DDoS attacks in the flying ad-hoc network (FANET), Mowla *et al.* [28] developed a model-free Q-learning mechanism with an adaptive exploration-exploitation epsilon-greedy policy, directed by an ondevice federated jamming detection mechanism. For DDoS attacks in the terrestrial network, Virupakshar *et al.* [29] used a variety of machine learning techniques to propose a network traffic monitoring system based on OpenStack firewall and raw socket programming. Alsirhani *et al.* [30] proposed a dynamic DDoS attack detection system based on classification algorithms, distributed systems, and fuzzy logic systems.

These studies have done a good job for DDoS detection in satellite networks or terrestrial networks. However, their research on NIDS is mainly focused on a part of the STIN domain, and few studies discuss how to establish NIDS from a global perspective to detect DDoS attack. The research on security data sets and FL makes NIDS adapt to the distributed architecture of STIN to achieve lower resource usage and higher classification accuracy. Therefore, we recommend using FL in STIN to implement distributed NIDS to ensure the privacy and security of its communication and information.

III. STIN SECURITY DATA SET

In this section, we will introduce the testbed architecture of STIN environment and the generation of security data set details in each domain.

A. DESCRIPTION OF TESTBED ARCHITECTURE

As Fig. 1 shows, we implement the testbed in a prototype and create hosts using Kernel-based Virtual Machine (KVM) [31] and OpenStack [32]. With the help of network virtualization technology, all these hosts in the testbed are implemented in eight high-performance servers, including one control

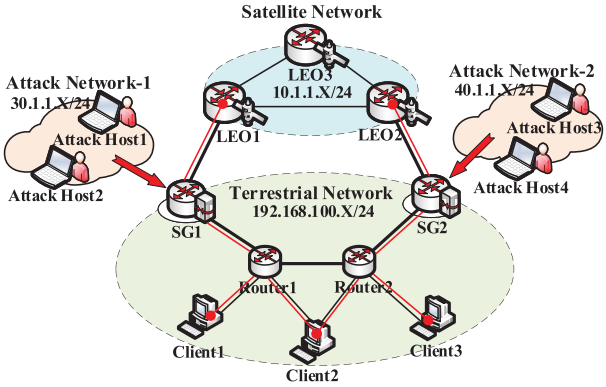

FIGURE 1. Testbed architecture.

node and seven computing nodes. The servers we use are DELL PowerEdge R720 rack-mount servers with Intel Xeon E5-2609 CPU, 32G memory and 1 TB hard-disk storage. Linux bridge is used to implement routing and switching between hosts. Next, we will introduce the testbed in three parts: satellite network, terrestrial network, and attack network.

1) SATELLITE NETWORK

Because this paper focuses on the identification of malicious traffic at the satellite network entrance, the scale of the satellite network is simplified in the testbed. We only use three Low Earth Orbit (LEO) satellites in the same orbit as test nodes. LEO1 and LEO2 remain connected to the satellite gateway1 (SG1) and SG2 of the terrestrial network, and the LEO satellites are always visible.

To make our testbed a real scenario, we simulate the actual latency and high-latency transmission protocols of the satellite network. We implement DTN in the satellite network to meet the need for high transmission delay with the help of Interplanetary Overlay Network (ION) [33], [34]. Then, we limit the transmission delay between LEO to 50ms and the transmission delay between LEO and SG to 3ms with the help of the Linux Traffic Control.

2) TERRESTRIAL NETWORK

As is shown in Fig. 1, the terrestrial network contains two SGs, two routers, and three clients. When deploying the terrestrial backbone network, considering that this paper focuses on designing security data sets based on traffic features, a simple star topology is used to avoid packet loss.

We use the TCP/IP protocol stack in the terrestrial network because it is used in the terrestrial networks in most cases. To facilitate protocol conversion between the satellite and terrestrial network, we develop a DTN-TCP/IP bidirectional conversion and deploy this function on two SGs. Moreover, the Tcpdump tool is installed on two SGs to capture the pcap files of the simulation uptime. Three clients are used to communicate with the satellite network to form benign traffic during the test time.

3) ATTACK NETWORK

In order to truly represent the modern threat environment, we installed different operating systems (Ubuntu14.04, Ubuntu16.04, Ubuntu18.04, and Win10) on the four attack hosts and set them in different network segments.

Since our proposed data set aims to test the attack detection technology of the STIN, it should cover a diverse set of network attack techniques and scenarios.

The terrestrial network has many security problems, and a DDoS attack is undoubtedly the most serious harm to it. Therefore, we hope that the malicious attacks on the terrestrial network can cover as many types as possible. We generated a total of nine different attack scenarios based on open source software simulations, including Botnets, Web Attacks, Backdoor, and six different DDoS attacks (LDAP, MSSQL, NetBIOS, Portmap, Syn, UDP).

Unlike terrestrial networks, satellite networks have a hig-her level of security and unique DTN transmission protocol, so malicious attacks applicable to terrestrial networks are difficult to enter into satellite networks and cause damage. However, DDoS attacks are flexible and changeable. Only by exploiting the loopholes in the DTN protocol's forwarding mechanism, it can greatly affect satellite communications. Since our prototype was implemented based on Interplanetary Overlay Network (ION) [33] software, we designed and implemented a DDoS attack for the DTN protocol, which is a cancellation transmission mechanism based on the LTP protocol. There is a loophole in the design of ION's LTP protocol, that is, after any party receiving and receiving receives Cancel_segment, it must return Cancel_ACK_segment in the direction of the connecting node that is communicating, without additional conditions. As soon as the node receives Cancel_segment, it must return a confirmation message. As shown in Fig. 2(a), the satellite network has a high resource occupancy rate during communication under normal conditions. When the attacking node launches an attack on the satellite network, as shown in Fig. 2(b), the attacker's control host will send a large number of Cancel_segments to the SG, it will force ION to reply to Cancel_ACK_segment to the party that normally communicates, occupying the bandwidth and computing resources of the link, thereby implementing the DDoS attack for the satellite network. Because the DTN transmission mechanism is more complex, only two types of satellite network DDoS attacks based on Syn and UDP are constructed.

B. SECURITY DATA SET DESIGN

Our proposed STIN security data set aims to collect various types of attacks in modern satellite and terrestrial network environments. We construct the data set based on the test architecture in Fig. 1. Meanwhile, to fully restore the modern threat environment, we set the experiment duration to 3 hours, starting at 15:00 and ending at 18:00. During the experiment, we kept each domain Transmission of benign traffic. As shown in Table 1, during the experiment, we performed

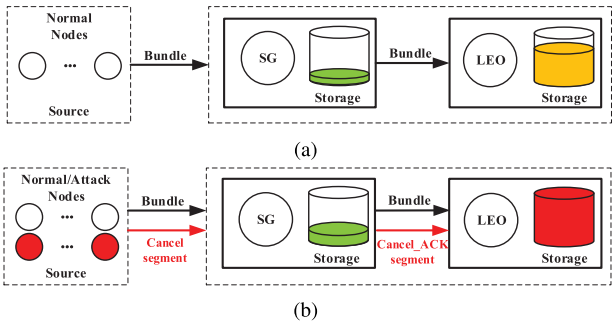

FIGURE 2. (a) Normal traffic in satellite networks. (b) DDoS attacks in satellite networks.

TABLE 1. Data set statistics.

Domain	Attacks	Attack Times
Terrestrial Network	Botnet	15:01 → 15:10
	Web Attack	15:21 → 15:31
	Backdoor	15:41 → 15:52
	LDAP_DDoS	16:01 → 16:11
	MSSQL_DDoS	16:21 → 16:30
	NetBIOS_DDoS	16:41 → 16:50
	Portmap_DDoS	17:01 → 17:13
	Syn_DDoS	17:21 → 17:32
	UDP_DDoS	17:41 → 17:52
Satellite Network	Syn_DDoS	15:23 → 15:57
	UDP_DDoS	16:52 → 17:20

nine types of terrestrial network attacks and two types of satellite network DDoS attacks.

Argus tool processes raw network packets (e.g., pcap files) and generates attributes/features of the network flow packets. CICFlowMeter is a flow-based feature extraction tool that can extract 80 functions in a pcap file. Finally, the output files of two different tools, Argus and CICFlowMeter, are stored in the Mysql database and the available features are filtered from them.

These features include packet-based features and flow-based features. Since most of the attacks we collect come from DDoS attacks, packet-based features are the most direct and fine-grained features, but they will greatly affect the processing efficiency in the actual analysis of network attacks. Therefore, we use the flow-level features available from the database to build the STIN security data set.

In order to adapt to the limited resources of satellite nodes, we should reduce the cost of training and testing as much as possible. Therefore, it is necessary to perform feature selection on the data set to retain the features with good training effects. First, we remove features related to the environment and time of the data set collection, such as five-tuples. Then, a large number of features with missing values or all zeros were deleted. Finally, Pearson coefficient and Variance selection are used to further extract features. Pearson coefficient is used to judge the correlation between feature columns, and the feature space is reduced by deleting the feature columns with strong correlation. Variance selection is used to filter the

TABLE 2. STIN data set features.

#	Name	Description
1	fl_dur	Flow duration
2	fw_pk	Total packets in the forward direction
3	l_fw_pkt	Total length of forward packets
4	l_bw_pkt	Total length of backward packets
5	pkt_len_min	Minimum length of a flow
6	pkt_len_max	Maximum length of a flow
7	pkt_len_std	Standard deviation length of a flow
8	fl_byt_s	Packet bytes transmitted per second
9	bw_iat_tot	Total time between of two backward packets
10	bw_iat_min	Minimum time between of two backward packets
11	fw_hdr_len	Number of bytes used in forward packet header
12	bw_pkt_s	Number of backward packets per second
13	syn_cnt	Number of packets with SYN
14	urg_cnt	Number of packets with URG
15	bw_win_byt	Number of backward bytes in the initial window

variance value of each column of features, and eliminate feature dimensions with smaller variance. As shown in Table 2, we finally screened out 15 best flow-level features that are helpful for intrusion detection to describe various types of benign and malicious traffic simulated in the prototype.

Based on the above-mentioned traffic extraction tools, we can collect about 5 million traffic every 10 minutes. Obviously, satellite nodes cannot afford such a huge amount of data. Therefore, we adopt a systematic sampling method to obtain a subset suitable for the processing capabilities of satellite nodes at a ratio of 1:500. Through filtering, we obtained 177,244 terrestrial network data and 132,320 satellite network data. The above two datasets were named TER20 and SAT20, and were stored in CSV format as the basis for designing the distributed NIDS. The source code and data of our data sets can be found at https://github.com/kun9717/STIN-data-set/.

IV. FL ADAPTED STIN ALGORITHM DESIGN AND IMPLEMENTATION

In this section, we first explain why the method of horizontal FL is used in a STIN. Then, we propose a FL adapted STIN algorithm to combine the horizontal FL method with a NIDS in STINs. After that, we propose a satellite network topology optimization algorithm that can be combined with NIDS and deployed in a prototype system.

For the sake of clear description, we make the definitions as shown in Table 3.

A. HORIZONTAL FL

The horizontal FL method means that when two data sets have more user features overlapping and their users overlap less, we divide the data set according to the horizontal (user dimension). Finally, we extract data with the same user characteristics but different users for training the neural network.

We define the matrix D to represent the data held by each data owner i. The rows of the matrix represent samples and the columns represent the features corresponding to the samples. We denote the feature space as X, the label space as Y and we

TABLE 3. Definition of the notations.

Notations	Description
T^s_{train}	The time duration of the satellite network training model
T^t_{train}	The time duration of the terrestrial network training model
T_{limit}	The limited-time of the training model
C^{init}_{model}	The complexity of initial FL model structure
C^{out}_{model}	The complexity of output FL model structure
N_{IGSO}	The number of satellites in IGSO layer
N_{MEO}	The number of satellites in MEO layer
W	Satellite connection weight matrix
$J_{connection}$	Satellite topology connection matrix

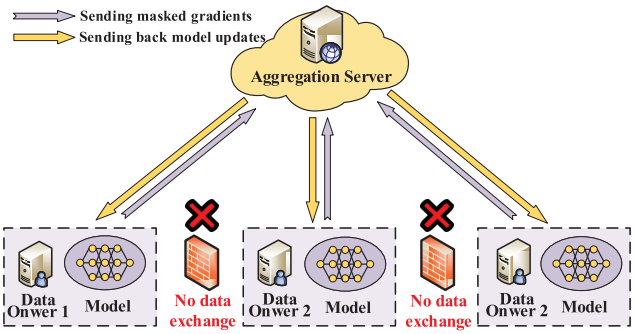

FIGURE 3. Architecture for a horizontal FL system.

use I as the sample ID space. From this, feature X, label Y and sample Ids I form a complete training data set (I, X, Y). In general, the horizontal FL is summarized as:

$$X_i = X_j,\ Y_i = Y_j,\ I_i \neq I_j,\ \forall D_i,\ D_j,\ i \neq j. \tag{1}$$

Fig. 3 presents the basic framework of horizontal FL in the networking environment. In this system, multiple data centers with the same data structure jointly learn a machine learning model with the support of a high-performance aggregation server. There is a huge barrier between data centers, so they can only rely on aggregation servers for data integration and transmission. At the same time, the server is honest, so it will not leak information to participants. Each participant trains the model locally and calculates its gradient, and transmits the gradient to the aggregation server through various network encryption technologies. After the aggregation server receives the data from each participant, it performs security aggregation without decryption, and feeds it back to each participant again to implement the gradient update of the local model.

Homomorphic Encryption (HE) is considered a viable approach to achieve secure multi-party computing in FL, so we would like to implement gradient encryption using the Paillier encryption system, which is an additive homomorphic public key encryption system. Given that Stochastic Gradient Descent (SGD) and its series of deformations are the most common deep learning optimization algorithms, we naturally think of building a federated optimization algorithm based on the parallel restarted SDG. Besides, we hope to use the Convolutional Neural Network (CNN) as the local neural network model to achieve higher recognition accuracy.

However, there are still many problems with deploying a lateral FL method in a STIN. On the one hand, satellite nodes have worse processor performance than ground-based computers, so they take longer to update the same machine learning model. In general, to ensure the completeness of data, the aggregation server will wait for all data centers to upload data before data integration. However, while the aggregation server is waiting for satellite nodes to upload data, the terrestrial network is constantly occupying the interface to monitor the data return from the aggregation server, which greatly wastes the resources of the terrestrial network. On the other hand, to combine NIDS with FL, it is necessary to overcome the long delay caused by FL in processing data. NIDS should be time-efficient and highly accurate, but machine learning methods generally require longer processing times. At the same time, the limited processing capacity of satellite nodes also leads to further extension of processing time, which is not conducive to timely analysis of various types of network traffic. Therefore, we need a FL adapted STIN algorithm to combine the horizontal FL method with a NIDS in STINs.

B. FL ADAPTED STIN ALGORITHM

Considering the key points mentioned in the previous section, we need to propose a FL adapted STIN algorithm to combine NIDS and FL in a STIN.

The FL adapted STIN algorithm needs to consider two key points, synchronization of processing time and efficiency of identifying traffic. Due to the limited resources of the satellite network, the complexity of the training model C_{model} will greatly affect the T_{train} of the satellite nodes, calculated as shown in (2).

$$C_{model} \sim O\left(\sum_{l=1}^{D} M_l^2 \cdot K_l^2 \cdot P_{l-1} \cdot P_l\right) \tag{2}$$

where D is the number of convolutional layers of the neural network, M is the side length of each convolution kernel output feature map, K is the side length of each convolution kernel, l is the lth convolutional layer of the neural network, P is the number of convolution kernels of the lth convolution layer.

Therefore, the purpose of the algorithm is to output a suitable C^{out}_{model} to achieve time synchronization in the FL process. Besides, we do not pay attention to the specific relationship between model complexity C_{model} and training time T_{train}. For this reason, we simply define that the training time T_{train} of each network is directly proportional to the model complexity C_{model}.

The basic procedure of FL adapted STIN algorithm is shown in Algorithm 1, which has the algorithm complexity of $O(n)$. The process of FL adapted STIN algorithm working out C^{out}_{model} values can be divided to two stages:

Algorithm 1 FL Adapted STIN Algorithm

Input: T^s_{train}, T^t_{train}, T_{limit}, C^{init}_{model}.
Output: C^{out}_{model}.

for masked gradients arrive **do**
  if $T^t_{train} \times 80\% \leq T^s_{train} \leq T^t_{train}$ **then**
    if $T^s_{train} \leq T_{limit}$ **then**
      $C^{out}_{model} = C^{init}_{model}$;
    else
      $C^{out}_{model} = C^{init}_{model}/[T^s_{train}/T_{limit}]$;
    end if
  else if $T^s_{train} \leq T^t_{train} \times 80\%$ **then**
    if $T^s_{train} \leq T_{limit}$ **then**
      $C^{out}_{model} = C^{init}_{model} \times [T^t_{train}/T^s_{train} \times 80\%]$;
    else if $T_{limit} < T^s_{train}$ **then**
      Reduce network complexity, rerun step 1;
    end if
  else if $T^t_{train} < T^s_{train}$ **then**
    if $T^s_{train} \leq T_{limit}$ **then**
      $C^{out}_{model} = C^{init}_{model}/[T^s_{train}/T^t_{train}]$;
    else
      if $T^t_{train} \leq T_{limit}$ **then**
        $C^{out}_{model} = C^{init}_{model}/[T^s_{train}/T_{limit}]$;
      else
        Reduce network complexity, rerun step 1;
      end if
    end if
  end if
end for
return C^{out}_{model};

1) TIME SYNCHRONIZATION DECISION

We compare the values of T^s_{train} and T^t_{train} to analyze the speed of satellite and terrestrial networks uploading data. We first set 80% as a threshold to allow satellite nodes to fluctuate in T^s_{train} values due to insufficient performance. The aggregation server receives the model training times T^s_{train}, T^t_{train} from the satellite and the terrestrial network, limited training time T_{limit} (varies with network size), and the initial model C^{init}_{model} as input. When masked gradients reach the aggregation server, we start to compare the values of T^s_{train} and T^t_{train}, and use it to determine whether the models transmitted by satellite network and terrestrial network are synchronized. If so, we judge whether T^s_{train} is within the specified time limit T_{limit}. If this condition is met, set C^{out}_{model} to C^{init}_{model} and output the result. If the conditions are not met, skip to step 2 to optimize efficiency. If not, we need to determine whether T^s_{train} is longer or shorter than T^t_{train}. If both T^s_{train} and T^t_{train} are higher than T_{limit}, there is a problem with the size of the STIN. We need to readjust the number of nodes or link connections in the network and rerun the algorithm. Otherwise, skip to step 2 to execute the efficiency decision.

2) EFFICIENCY JUDGMENT

An important basis for optimizing efficiency is to simplify the complexity of the model to reduce the training time of the satellite network, so that the network eventually achieves higher data processing efficiency. Therefore, we need to process the value of C^{init}_{model}.

We compare the sizes of T^s_{train}, T^t_{train}, and T_{limit} to decide whether to increase or decrease the complexity of the model. The C^{out}_{model} conversion formula and decision conditions have been given in Algorithm 1. We can use this to obtain C^{out}_{model} as output. The final model C^{out}_{model} can not only save the resource occupancy rate of different STINs, but also meet the time-efficient requirements of NIDS.

C. PROTOTYPE IMPLEMENTATION

NIDS based on DL or FL relies heavily on a relatively stable network topology, which has a clear transmission path and continuous traffic. However, traditional satellite networks will always switch topologies, which causes great problems for tracing attack sources and analysising features. Therefore, in this section, we adopt the satellite network topology optimization algorithm proposed in the previous work [35] to establish a stable topology when the number of antennas is limited. In the end, we combined this topology with NIDS using FL and implemented them in a prototype system.

Different from terrestrial networks, satellite networks have time-varying characteristics regardless of nodes or topologies. However, the satellite network has the characteristics of predictability and periodicity. At the same time, the movement of the satellite node has a slow and insignificant effect on the continuous satellite time-varying topology network G. Therefore, based on the knowledge of graph theory, the time-varying satellite network topology in a continuous state is static within a period, and the model of defining the satellite network topology is: $G(V, A, W)$.

Where $V = \{v_1, v_2 \ldots v_n\}$ represents the set of finite nodes in the satellite network; A represents a finite set of links, $A \subseteq V \times V$; $W = \{w_{ij}\}$ represents a weight matrix, and w_{ij} represents the stability weight between node v_i and v_j, which is defined as:

$$w_{ij} = \begin{cases} \log_2\left(\frac{T_{period}}{T_{ij}} \cdot \frac{L_{ij}}{L_{\min}} \cdot \frac{D_{ij}}{D_k}\right), & i = j \\ 0, & i \neq j \end{cases} \tag{3}$$

where T_{period} is the period of the satellite topology network,T_{ij} is the maximum visible time of satellite nodes v_i and v_j within a period, L_{ij} is the inter-satellite link capacity, and the minimum link capacity L_{min} is required to prevent network congestion; The satellite node has its own link weight of 0, that is, it is impossible to establish a link with itself; $k = \{1, 2\}$ represents the intra-layer or inter-layer links of the satellite, and different link distance thresholds D_k should be defined for links at different levels. D_{ij} is the weighted distance between satellite node v_i and v_j within one period T_{period}, which is defined as follows:

$$D_{ij} = \frac{T_s}{T_{period}} \cdot \sum_{m=1}^{n} D^{ij}_m \tag{4}$$

Algorithm 2 Satellite Network Topology Optimization Algorithm

Input: N_{IGSO}, N_{MEO}, W.
Output: $J_{connection}$.

```
for i = 1 to N_IGSO do
    GEO satellites are connected in pairs as an alternative path, and all elements in row i and column i of M_connection are set to 1;
    for j = 1 to N_MEO do
        Select a column with element 1 in the (j + N_IGSO)th row of W, record it as k;
        J_ij^connection = J_ji^connection = 1;
        w_ij = w_ji = 0;
    end for
end for
return J_connection;
```

TABLE 4. Parameters of IGSO/MEO constellation.

Constellation parameter	IGSO	MEO
Inclination (°)	55	55
Altitude (km)	35768	21500
Number of planes	3	3
Number of satellites per plane	1	8
Number of antennas	3	2-3

where $s = \{t_1, t_2 \ldots t_n\}$ represent time slices for visibility between satellite nodes, T_s is the length of each time slice, and D_m^{ij} is the physical distance between node v_i and v_j within each time slice physical distance. Therefore, a weighted distance is assigned to each inter-satellite link as the basis for measuring link connection quality to evaluate the impact on link stability.

The basic procedure of the satellite network topology optimization algorithm is shown in Algorithm 2. The algorithm first refers to the link weights to establish a stable topology in the Inclined Geo Synchronous Orbit (IGSO) layer, and then we fully consider the connection weight W in the Medium Earth Orbit (MEO) layer to establish a stable double-layer satellite network topology. $J_{connection}$ represents the connection status of the nodes in the satellite network as output.

In this paper, the standard Walker24/3/1 constellation is adopted, which consists of 24 MEO satellites and 3 IGSO satellites form a double-layer satellite network. We implement DTN in the satellite networks to meet the need for high transmission delay with the help of ION. Table 4 shows the specific parameters of the IGSO/MEO constellation.

In order to describe a sufficiently large STIN, we also deployed 3 terrestrial domains in the prototype, each of which includes 4 hosts, 2 servers, and 3 routers. As shown in Fig. 4, the nodes are connected by the tree topology which is widely used in the construction of terrestrial networks. We verified that the network complexity of the proposed STIN meets the requirements of the FL adapted STIN algorithm.

The distributed NIDS includes two stages of offline training and traffic monitoring. During the offline training phase, we deploy machine learning models in two SGs and two Edge Routers. Among them, Edge Router 1 both stores local training data and acts as an aggregation server to aggregate data from all parties, thus simplifying the data transfer process in FL. We input the SAT20 to the SGs and the TER20 to the Edge Routers. Machine learning models in different nodes transmit their training gradients to Edge Router 1 after their learning. Then, server send them back to each node after encryption and aggregation. During the traffic monitoring phase, each SG and Edge Router can independently monitor the traffic coming from the external network and determine whether it is malicious traffic. If so, add the corresponding firewall policy and block the traffic. Otherwise, temporarily add it to the firewall whitelist and trust the data traffic from the source address in the short term.

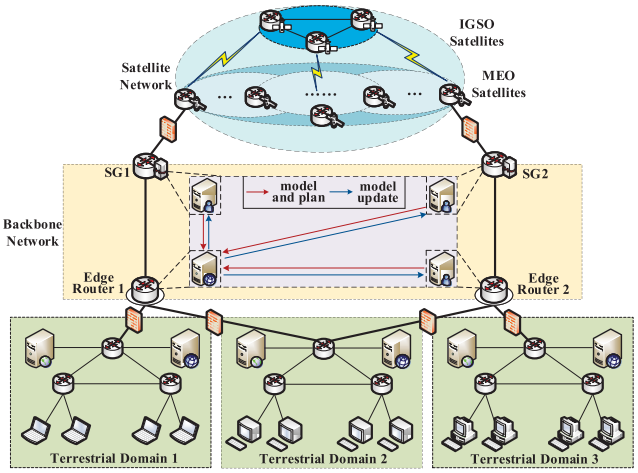

FIGURE 4. Illustration of the prototype.

V. PERFORMANCE EVALUATION

In this section, the performance of our proposed distributed NIDS is studied in an emulation environment based on a Linux-based prototype. We first introduce the STIN topology used in our emulation and the associated emulation parameters. Then, we propose the evaluation criteria for performance evaluation. Finally, we describe the performance evaluation of distributed NIDS using FL in a STIN.

A. EMULATION SETTINGS

Our emulation is based on the OpenStack and TensorFlow Federated [36] (version 2.2.0, running on CPU), which is an open-source framework for machine learning and other computations on decentralized data. We use the prototype shown in Fig. 4 to implement distributed NIDS, which is also deployed on the OpenStack platform where the testbed is located. All nodes are implemented using virtualization technology, and they implement routing and switching through Linux bridges. Among them, each link of the terrestrial network and the backbone network is set to 100 Mbps. The link delay parameters of the STIN are shown in Table 5.

CNN has the sparsity of connections, so each of its regions has its own unique characteristics, which complements the distributed characteristics of FL. Thus, deep CNN is used

TABLE 5. Parameters of link delay.

	IGSO satellites	MEO satellites	SGs	Edge routers
IGSO satellites	\	86 ms	\	\
MEO satellites	86 ms	\	50 ms	\
SGs	\	50 ms	\	3 ms
Edge routers	\	\	3 ms	\

TABLE 6. Main parameter values of Sec.IV.

Parameter	Value	Parameter	Value
T_{period}(s)	86400	N_{MEO}	24
$C_{\min}$(Mbps)	500	K_1	5
D_1(km)	66000	K_2	3
D_2(km)	40000	C_0	1
T_s(s)	60	C_1	32
N_{IGSO}	3	C_2	64

as the deep learning model deployed in FL. Based on the FL adapted STIN algorithm, we need to reasonably control the complexity of the CNN structure under the established topology to achieve higher data processing efficiency.

The SAT20 and TER20 are divided into 80% training set and 20% testing set. Two training sets are stored in the SGs and Edge Routers respectively for local model training. Two testing sets are combined into a comprehensive testing set and stored in a aggregation server to evaluate the performance of the final model.

Finally, we set T_{limit} to 10 seconds. The values of T^s_{train}, T^t_{train} and M_l all change with the structure or input data of CNN. For the neural network in FL, CNN consists of two convolutional layers, two pooling layers and one dropout layer. The optimization algorithm is based on the parallel restarted SDG.

Table 6 shows the values of the main parameters in Section IV.

B. EVALUATION CRITERIA

There are different evaluation criteria commonly used to determine the performance of NIDSs or FL. In this section, the evaluation criteria used for performance evaluation are explained below.

- Time cost: The total time cost of FL is the sum of local training time and communication time in all communication rounds, calculated as shown in (5).

$$T_{total} = \sum_{i=1}^{rounds} (T_{local} + T_{communication}). \tag{5}$$

  where T_{local} is equal to $max(T^s_{train}, T^t_{train})$, which varies with the size of the data set and C^{out}_{model}, $T_{communication}$ represents the time to complete a round of model transmission and update between the Edge Router and the SG, which is limited by the upload bandwidth in the STIN.

- Accuracy: Accuracy value is the ratio of the number of samples correctly classified by the system to the total number of samples, calculated as shown in (6).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{6}$$

  where True-Positive (TP) is the number of samples correctly predicted in the intrusion class; True-Negative (TN) is the number of samples correctly predicted in the normal class; False-Negative (FN) is the number of samples incorrectly predicted in the normal class; False-Positive (FP) is the number of samples incorrectly predicted in the intrusions class.

- False Positive Rate (FPR): FPR is the ratio of malicious traffic being misidentified as normal traffic to the total number of malicious traffic in NIDS, calculated as shown in (7).

$$FPR = \frac{FP}{TN + FP}. \tag{7}$$

- Training loss: We use the mean square error (MSE) to estimate the expected value of the square of the difference between the predicted value $f(X)$ and the true value Y. MSE can evaluate the degree of data change. The smaller the value of MSE, the better accuracy of the training model. It is calculated as seen in (8).

$$MSE = \frac{1}{n} \cdot \sum_{i=1}^{n} (Y - f(X))^2. \tag{8}$$

  where $f(X)$ represents the predicted value output by the training model with C^{out}_{model} after each round of FL training, Y represents the true label in SAT20 or TER20.

- CPU utilization: CPU utilization refers to the percentage of CPU occupied in real time during program operation.
- Packet loss rate: Packet loss rate is the ratio of the number of lost data packets to the transmitted data group.
- Malicious traffic recognition rate: Malicious traffic reco-gnition rate is the ratio of malicious traffic that NIDS recognizes from the total incoming traffic, calculated as shown in (9).

$$Recognition\ rate = \frac{Flow_{malicious}}{Flow_{input}} \times 100\%. \tag{9}$$

C. PERFORMANCE ANALYSIS

We designed three sets of experiments, including minimizing the processing time of FL, comparing the effects of FL and traditional DL in STIN, and comparing the performance differences between distributed NIDS and traditional IDS.

1) TIME COST OF FL

In terms of FL in a general environment, due to the rapid development of the Graphics Processing Unit (GPU), the computational cost is lower compared to the communication cost. Therefore, if additional calculations are added to each node to reduce the number of communication rounds required to train the model, the time cost can be reduced.

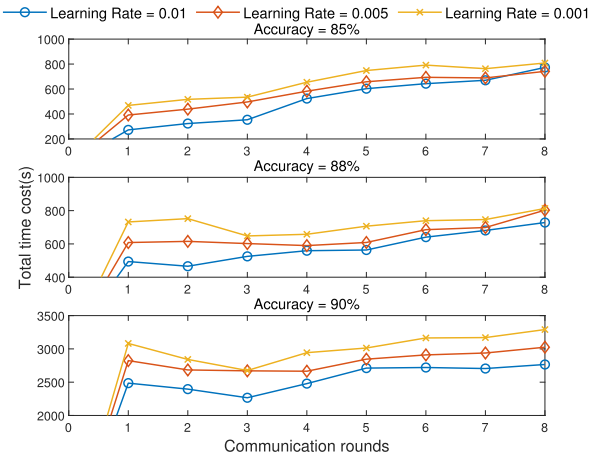

FIGURE 5. Total time cost of FL communication rounds.

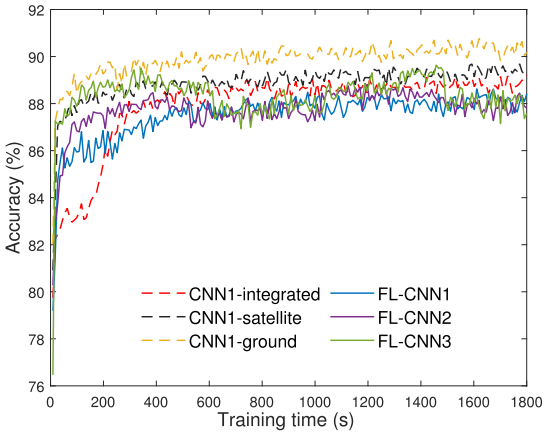

FIGURE 6. Accuracy of training time.

However, the computing resources of satellites are much lower than that of the terrestrial calculator. Therefore, regardless of T_{local} or $T_{communication}$, it will affect the processing time of FL in the STIN.

We need to find a balance between the training round of the local model and the communication round of the training model. Therefore, we use a certain numerical Accuracy as a standard to measure the total communication cost of FL. It is calculated as seen in (6).

Because our goal is to evaluate to minimize the time cost of federated learning, rather than to achieve the best accuracy on this task, we set target accuracy rates of 85%, 88%, and 90%, respectively. Besides, learning rate is an important hyperparameter in deep learning, which determines whether the objective function can converge to a local minimum and when it converges to the minimum. Therefore, to avoid the influence of the learning rate on the experimental results, we respectively compared the influence of different learning rates of the model on the time cost.

According to (5) and (6) above, Fig. 5 shows the relationship between the communication rounds of FL and the total time cost under different target accuracy. It can be observed that when the accuracy requirement is low, the fewer communication rounds, the lower the time cost; when the accuracy requirement gradually increases, it is necessary to appropriately increase the communication rounds to reduce the total time cost. Taking the target accuracy rate of 90% as an example, we choose 3 communication rounds will reduce the total time cost by 8.36% compared with other choices. In addition, different learning rates also conform to the law of communication rounds and time cost.

2) FLOW RECOGNITION ACCURACY OF DIFFERENT DL

After adjusting the communication rounds of FL, we will test the performance of FL offline training. As mentioned in the previous section, we have divided the SAT20 and TER20 into training sets and testing sets. In addition, we deployed three DL methods for comparison. Each model is described as follows:

CNN1: We use the same model structure as the deep CNN in FL to compare performance.

CNN2: Compared with CNN1, a convolutional layer and a pooling layer are added.

CNN3: We doubled the model of CNN1, adding 2 convolutional layers and 2 pooling layers.

Since the focus of FL is to solve the problem of data islands, it has not improved the learning model. Therefore, we use three CNN1s with the same structure as the CNN in FL, and evaluate the efficiency of FL through different training sets and testing sets. CNN1-integrated combined the SAT20 and TER20 into a comprehensive data set as the training set and the testing set; CNN1-satellite used the SAT20 as the training set and the TER20 as the testing set; CNN1-ground used the TER20 as the training set, and the SAT20 is used as the testing set.

Besides, due to the high complexity of the models of CNN2 and CNN3, resulting in $T^s_{train} > T^t_{train} > T_{limit}$, they do not satisfy the FL adapted STIN algorithm. Therefore, we will also deploy CNN2 and CNN3 in FL for comparison. We select the training model in Edge Router 1 to participate in the evaluation.

Fig. 6 and Fig. 7 show the accuracy and loss rate of three FL and three CNN1s in the training time within half an hour, which have been demonstrated by (6) and (8). It can be seen from Fig. 6 that in most cases, CNN1-integrated has a higher recognition accuracy rate than FL. The reason is that CNN1-integrated has no model changes caused by information exchange between hosts. CNN1-satellite and CNN1-ground show better performance in Fig. 6 and Fig. 7 because their training data is only traffic information in a single network environment, with fewer data entries and types of attacks, it is easier to identify. In addition, compared to CNN2 and CNN3, the CNN1 model deployed in FL shows high stability during training time. This is because the CNN1 structure conforms to FL adapted STIN algorithm, which reduces the training time in the satellite network so

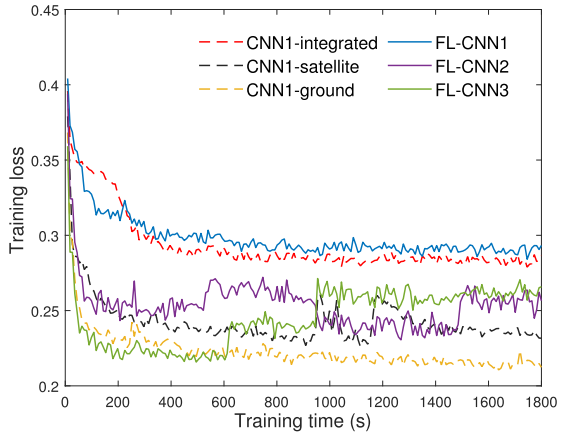

FIGURE 7. Training loss of training time.

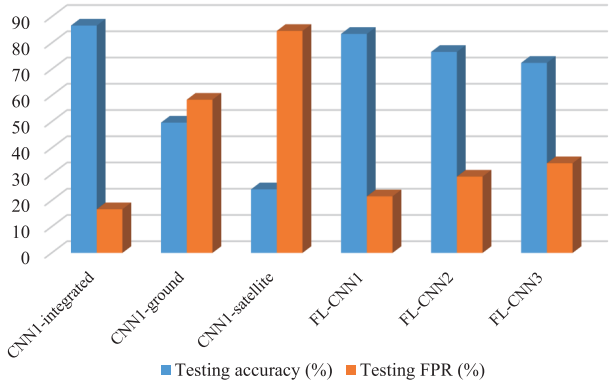

FIGURE 8. Testing accuracy and FPR of six methods.

that the STIN achieves higher synchronization of processing time.

According to the calculation of (6) and (7), Fig. 8 shows the testing accuracy and FPR of each method on the respective testing set. Although CNN1-ground and CNN1-satellite showed the highest accuracy during the training process, they showed lower testing accuracy and extremely high testing FPR when verifying the model through the testing set. This shows that the attack data in a heterogeneous network is very different. CNN1-integrated has the highest testing accuracy and the lowest testing FPR among the six methods. The reason is that FL does not improve the learning model. Therefore, in the case of a centralized data set, the traditional DL method can show better classification performance. However, in a STIN, data is restricted by security and privacy, and it is difficult to combine them. Therefore, FL is a suitable solution. In the comparison of the three federated learning models, the FL-CNN1 model adopted in this paper achieves the highest testing accuracy and the lowest testing FPR, thanks to the implementation of the model based on the FL adapted STIN algorithm. The models of FL-CNN2 and FL-CNN3 are too complicated for resource-constrained satellite nodes, so the terrestrial node cannot complete the training synchronously with the satellite node, resulting in lower data processing efficiency and higher testing FPR.

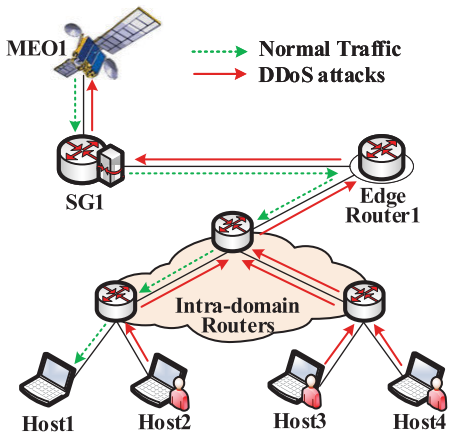

FIGURE 9. Attack scenario.

TABLE 7. Terrestrial flow type and time.

Flow type	Syn_DDoS	UDP_DDoS	5MB/s files	Slowloris
Beginning and ending times	600s–660s	900s–960s	1200s–1800s	2700s–2760s

3) PERFORMANCE COMPARISON OF DIFFERENT NIDS

Through the above work, we have designed a suitable FL framework for distributed NIDS. Next, we will compare the performance of three distributed NIDSs using FL with traditional IDSs in the same intrusion detection environment, including Snort [37] and Bro IDS [38], to verify the effect of the distributed NIDS using FL.

In order to emulate the normal traffic environment, we used the MEO1-SG1-Edge Router1-Intra-domain Routers-Host1 path to send a 1MB text file per second from the MEO node to the terrestrial node for a total of 1 hour. he attack scenario is shown in Fig. 9. Among them, MEO1 is the MEO node directly connected to SG1, and Host1 is a host in terrestrial domain 1. We use Hping3 and Python tools to emulate DDoS attacks, and finally three other hosts in the terrestrial domain 1 attack MEO1. The types and beginning and ending times of DDoS attacks are shown in Table 7.

We have deployed FL-CNN1 + Iptable [39], FL-CNN2 + Iptable, FL-CNN3 + Iptable, Snort + Iptable, and Bro IDS five NIDSs and processing tools on the Edge Router1. Among them, Snort uses its own detection rules, and Bro IDS writes the scripting language based on Snort rules. Fig. 10 shows the throughput changes of the egress router of terrestrial domain1 during the emulation time under five intrusion detection scenarios. Each intrusion detection scenario has experienced two DDoS flood attacks, a slow attack, and a large file transfer. The changing trend of their input flow is basically the same.

As shown in Fig. 11, we calculated the recognition rates of five types of NIDS for different malicious traffic according to (9). For the traditional Syn_DDoS and UDP_DDoS, the five types of NIDS show similar recognition capabilities. For short-term burst traffic and Slowloris attacks, distributed

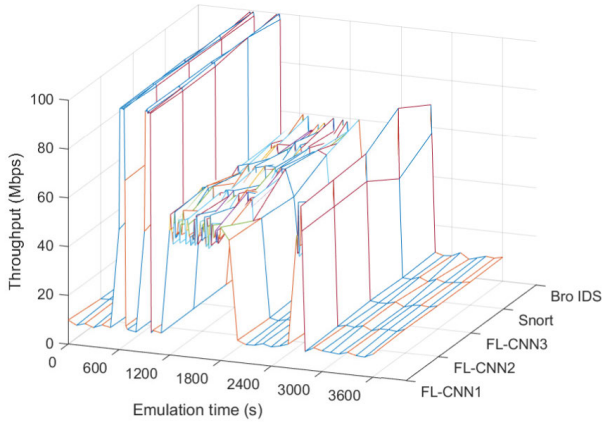

FIGURE 10. Throughput of the egress router of terrestrial domain1.

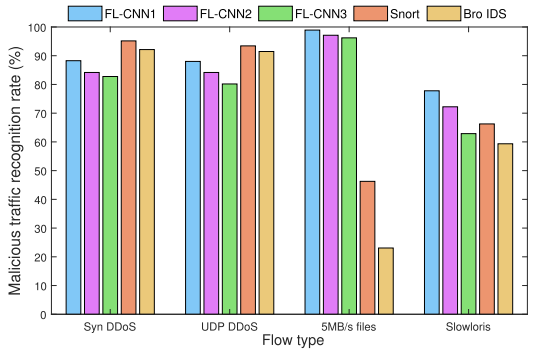

FIGURE 11. Malicious traffic recognition rate of five NIDSs.

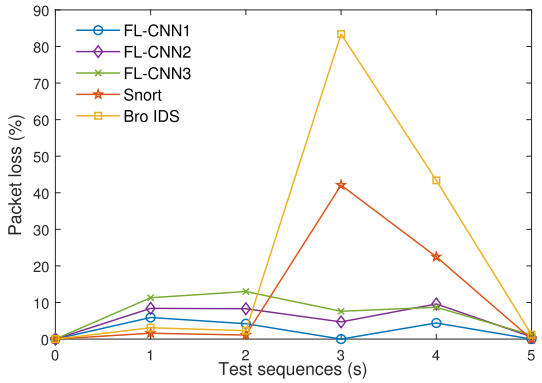

FIGURE 12. Packet loss of five NIDSs.

NIDS using FL show better performance than traditional IDS. The reason is that distributed NIDS using FL can distinguish whether it is a network attack based on the characteristics of the traffic, rather than a simple five-tuple. Therefore, it has a certain ability to identify normal traffic or malicious attacks that are not involved in training data sets. Besides, due to the C_{model} of CNN1 adapted to STIN, the distributed NIDS using FL-CNN1 exhibits a higher rate of malicious traffic recognition than other FL models.

We use iPerf [40] to measure the packet loss rate of normal traffic on the transmission path from MEO1 to Host1. Fig. 12 selects 0s, 630s, 930s, 1500s, 2730s, and 3600s to plot the

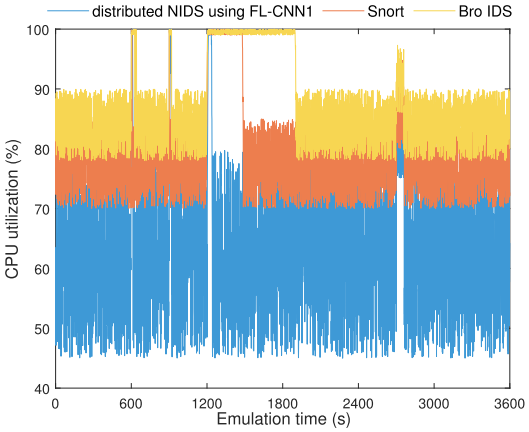

FIGURE 13. CPU utilization of Edge Router1.

packet loss rate under the five NIDSs at each moment. Fig. 12 shows that the distributed NIDS using FL-CNN1 can get near-optimal results, exceeding the result of the other NIDSs.

Through the above analysis, it can be known that FL-CNN1 shows the best performance compared with other CNN algorithms implemented in FL. Therefore, we use the distributed NIDS using FL-CNN1 as a representative to compare the CPU utilization with two traditional IDSs. Fig. 13 shows the CPU utilization of the Edge Router1 using three NIDSs. Compared with Snort and Bro IDS, distributed NIDS using FL-CNN1 not only exhibits lower CPU utilization in most cases, but also has the fastest response time under different DDoS attacks.

The emulation results in Fig. 11, Fig. 12, and Fig. 13 demonstrate that distributed NIDS using FL-CNN1 can adapt to resource-constrained STIN and identify malicious traffic more comprehensively.

VI. CONCLUSION

In this paper, we propose a distributed NIDS using FL in a STIN, aiming to meet the requirements for traffic security and privacy in heterogeneous networks. We propose FL adapted STIN algorithm and satellite network topology optimization algorithm to adapt to NIDS. We deployed the distributed NIDS in a prototype system and conducted extensive emulations to evaluate the performance of our approach by randomly launching malicious attacks. Related results demonstrate that compared with traditional deep learning and intrusion detection systems, the distributed NIDS using FL has a higher malicious traffic recognition rate, lower packet loss rate, and lower CPU utilization rate. Our future work will follow three directions. Firstly, we plan to increase the types of DDoS attacks from six to more than fifteen. We will extend our current work to the 5G mobile communication network access scenario with massive terminal equipment. Secondly, starting from the principles of various DDoS attacks, we are ready to construct a new flow feature extraction method and generate a unique malicious flow behavior data set. Finally, based on the knowledge graph, we will establish a distributed network behavior knowledge base in massive terminal

scenario to effectively respond to the endless threats and challenges brought by the 5G scenario.

REFERENCES

[1] T. Li, H. Zhou, H. Luo, I. You, and Q. Xu, "SAT-FLOW: Multi-strategy flow table management for software defined satellite networks," *IEEE Access*, vol. 5, pp. 14952–14965, 2017.
[2] S. Yu, Y. Tian, S. Guo, and D. O. Wu, "Can we beat DDoS attacks in clouds?" *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2245–2254, Sep. 2014.
[3] J. Shen, T. Zhou, D. He, Y. Zhang, X. Sun, and Y. Xiang, "Block design-based key agreement for group data sharing in cloud computing," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 6, pp. 996–1010, Nov./Dec. 2019.
[4] J. Shen, C. Wang, S. Ji, T. Zhou, and H. Yang, "Secure emergent data protection scheme for a space-terrestrial integrated network," *IEEE Netw.*, vol. 33, no. 1, pp. 44–50, Jan. 2019.
[5] W. Wu, R. Li, G. Xie, J. An, Y. Bai, J. Zhou, and K. Li, "A survey of intrusion detection for in-vehicle networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 919–933, Mar. 2020.
[6] H. Yao, P. Gao, P. Zhang, J. Wang, C. Jiang, and L. Lu, "Hybrid intrusion detection system for edge-based IIoT relying on machine-learning-aided detection," *IEEE Netw.*, vol. 33, no. 5, pp. 75–81, Sep. 2019.
[7] W. Zhong, N. Yu, and C. Ai, "Applying big data based deep learning system to intrusion detection," *Big Data Mining Anal.*, vol. 3, no. 3, pp. 181–195, Sep. 2020.
[8] S. Otoum, B. Kantarci, and H. T. Mouftah, "On the feasibility of deep learning in sensor network intrusion detection," *IEEE Netw. Lett.*, vol. 1, no. 2, pp. 68–71, Jun. 2019.
[9] H. Yang and F. Wang, "Wireless network intrusion detection based on improved convolutional neural network," *IEEE Access*, vol. 7, pp. 64366–64374, 2019.
[10] K. B. V., N. D. G., and P. S. Hiremath, "Detection of DDoS attacks in software defined networks," in *Proc. 3rd Int. Conf. Comput. Syst. Inf. Technol. Sustain. Solutions (CSITSS)*, Bengaluru, India, Dec. 2018, pp. 265–270.
[11] *NSLKDD*. Accessed: 2009. [Online]. Available: https://nsl.cs.unb.ca/NSLKDD/
[12] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Canberra, ACT, Australia, Nov. 2015, pp. 1–6.
[13] M. Hao, H. Li, G. Xu, S. Liu, and H. Yang, "Towards efficient and privacy-preserving federated deep learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Shanghai, China, May 2019, pp. 1–6.
[14] H. Wen, Y. Wu, C. Yang, H. Duan, and S. Yu, "A unified federated learning framework for wireless communications: Towards privacy, efficiency, and security," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Toronto, ON, Canada, Jul. 2020, pp. 653–658.
[15] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4177–4186, Jun. 2020.
[16] B. Feng, H. Zhou, H. Zhang, G. Li, H. Li, S. Yu, and H.-C. Chao, "HetNet: A flexible architecture for heterogeneous satellite-terrestrial networks," *IEEE Netw.*, vol. 31, no. 6, pp. 86–92, Nov. 2017.
[17] *Tcpdump*. Accessed: 2019. [Online]. Available: https://www.Tcpdump.org/
[18] *Argus*. Accessed: 2019. [Online]. Available: https://www.openargus.org/
[19] *CICFlowMeter*. Accessed: 2017. [Online]. Available: https://www.github.com/ISCX/
[20] F. Salo, M. Injadat, A. B. Nassif, A. Shami, and A. Essex, "Data mining techniques in intrusion detection systems: A systematic literature review," *IEEE Access*, vol. 6, pp. 56046–56058, 2018.
[21] *KDD99*. Accessed: 2007. [Online]. Available: https://kdd.ics.uci.edu/databases/
[22] T.-Y. Mu, A. Al-Fuqaha, and K. Salah, "Automating the configuration of MapReduce: A reinforcement learning scheme," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 50, no. 11, pp. 4183–4196, Nov. 2020.
[23] M. Langer, Z. He, W. Rahayu, and Y. Xue, "Distributed training of deep learning models: A taxonomic perspective," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 12, pp. 2802–2818, Dec. 2020.
[24] S. A. Rahman, H. Tout, C. Talhi, and A. Mourad, "Internet of Things intrusion detection: Centralized, on-device, or federated learning?" *IEEE Netw.*, early access, Sep. 1, 2020, doi: 10.1109/MNET.011.2000286.
[25] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, 3rd Quart., 2020.
[26] F. Li, L. Yin, and W. Wu, "Research status and development trends of security assurance for space-ground integration information network," *J. Commun.*, vol. 37, no. 11, pp. 156–168, 2016.
[27] A. Di, S. Ruisheng, L. Lan, and L. Yueming, "On the large-scale traffic DDoS threat of space backbone network," in *Proc. IEEE 5th Int. Conf. Big Data Secur. Cloud (BigDataSecurity), Int. Conf. High Perform. Smart Comput. (HPSC), IEEE Int. Conf. Intell. Data Secur. (IDS)*, Washington, DC, USA, May 2019, pp. 192–194.
[28] N. I. Mowla, N. H. Tran, I. Doh, and K. Chae, "AFRL: Adaptive federated reinforcement learning for intelligent jamming defense in FANET," *J. Commun. Netw.*, vol. 22, no. 3, pp. 244–258, Jun. 2020.
[29] K. B. Virupakshar, M. Asundi, K. Channal, P. Shettar, S. Patil, and D. G. Narayan, "Distributed denial of service (DDoS) attacks detection system for OpenStack-based private cloud," *Procedia Comput. Sci.*, vol. 167, pp. 2297–2307, Jan. 2020.
[30] A. Alsirhani, S. Sampalli, and P. Bodorik, "DDoS detection system: Using a set of classification algorithms controlled by fuzzy logic system in apache spark," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 3, pp. 936–949, Sep. 2019.
[31] *KVM*. Accessed: 2017. [Online]. Available: https://www.linux-kvm.org/
[32] *OpenStack*. Accessed: 2017. [Online]. Available: https://www.openstack.org/
[33] T. Li, H. Zhou, H. Luo, and S. Yu, "SERvICE: A software defined framework for integrated space-terrestrial satellite communication," *IEEE Trans. Mobile Comput.*, vol. 17, no. 3, pp. 703–716, Mar. 2018.
[34] B. Feng, G. Li, G. Li, Y. Zhang, H. Zhou, and S. Yu, "Enabling efficient service function chains at terrestrial-satellite hybrid cloud networks," *IEEE Netw.*, vol. 33, no. 6, pp. 94–99, Nov. 2019.
[35] K. Li, H. Zhou, T. Zhe, and G. Li, "SLSA: A link stability based algorithm for the topology optimization of IGSO/MEO double-layered satellite network," in *Proc. ReBICTE*, vol. 5, Nov. 2019, pp. 1–11.
[36] *TensorFlow Federated*. Accessed: 2020. [Online]. Available: http://www.tensorflow.google.cn/federated/
[37] *Snort*. Accessed: 2018. [Online]. Available: https://www.snort.org/
[38] *Bro IDS*. Accessed: 2018. [Online]. Available: https://www.bro.org/
[39] *Iptable*. Accessed: 2020. [Online]. Available: https://www.netfilter.org/
[40] *iPerf*. Accessed: 2020. [Online]. Available: https://iperf.fr/

KUN LI received the B.S. degree in telecommunications engineering from Beijing Jiaotong University (BJTU), China, in 2014, where he is currently pursuing the Ph.D. degree in information and communication engineering. He joined the National Engineering Laboratory for Next Generation Internet Interconnection Devices, BJTU. His research interests include the architecture of next generation internet, and network security. His research interests also include satellite networks and machine learning.

HUACHUN ZHOU received the B.S. degree from the People's Police Security University of China, in 1986, and the M.S. degree in telecommunication automation and the Ph.D. degree in telecommunications and information system from Beijing Jiaotong University (BJTU), in 1989 and 2008, respectively. In 1994, he joined the Institute of Automation Systems, BJTU, where he is currently a Lecturer. From 1999 to 2009, he was a Senior Engineer with the School of Electronic and Information Engineering, BJTU, and with the Network Management Research Center, BJTU. Since 2009, he has been a Professor with the National Engineering Laboratory for Next Generation Internet Interconnection Devices, BJTU. He has authored over 40 peer-reviewed articles. He holds 17 patents. His research interests include mobility management, mobile and secure computing, routing protocols, network management, and satellite networks.

ZHE TU received the B.S. degree in telecommunications engineering from Beijing Jiaotong University (BJTU), China, in 2014, where he is currently pursuing the Ph.D. degree in information and communication engineering. He joined the National Engineering Laboratory for Next Generation Internet Interconnection Devices, BJTU. His research interests include the architecture of next generation internet, network service management, and network security. His research interests also include satellite networks and mobile internet.

WEILIN WANG received the B.S. degree in telecommunications engineering from Beijing Jiaotong University (BJTU), China, in 2020, where she is currently pursuing the M.S. degree in communication and information system. She joined the National Engineering Laboratory for Next Generation Internet Interconnection Devices, BJTU. Her research interests include the architecture of next generation internet, network security, and machine learning.

HONGKE ZHANG (Senior Member, IEEE) received the Ph.D. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 1992. He is currently a Professor with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China, where he directs the National Engineering Laboratory on the Internet Technology for Next-Generation Internet, China. He is also the Chief Scientist of the National Basic Research Program of China (973 Program). His current research has resulted in many research articles, books, patents, systems, and equipment in the areas of communications and computer networks.