Protecting IoT Servers Against Flood Attacks with the Quasi Deterministic Transmission Policy

Erol Gelenbe
Institute of Theoretical & Applied Informatics
Polish Academy of Science, 44-100 Gliwice, PL
& CNRS I3S Lab., Université Côte d'Azur, FR
& Yaşar University, Bornova, Izmir, TR
ORCID:0000-0001-9688-2201

Mohammed Nasereddin
Institute of Theoretical & Applied Informatics
Polish Academy of Science, 44-100 Gliwice, PL
ORCID: 0000-0002-3740-9518

***Abstract*—Servers at Supply Chains that receive packets from IoT devices should meet the QoS needs of incoming packets, and protect the system from Cyberattacks. UDP Floods are often part of Cyberattacks that overwhelm Supply Chains and the IoT through congestion that paralyzes their operation and limits their ability for timely Attack Detection. Thus this paper proposes an architecture that protects a connected Server using a Smart Quasi-Deterministic Transmission Policy Forwarder placed at its input port. This Forwarder shapes the incoming traffic, sends it to the Server without modifying the overall packet delay, and avoids Server congestion. The relevant theoretical background is reviewed, and measurements during a UDP Flood Attack are provided to compare the Server performance, with and without the Forwarder. It is seen that during a UDP Flood Attack, the Forwarder protects the Server from congestion, allowing it to effectively identify Attack Packets. The resulting Forwarder congestion is also eliminated with "drop" commands generated by the Forwarder, or sent by the Server to the Forwarder.**

***Index Terms*—Intrusion Detection, Supply Chains, IoT, Traffic Shaping, Quasi-Deterministic Transmission Policy (QDTP)**

I. Introduction

With some 30 Billion devices on the Internet [1], many types of anomalies have been observed as a result of cyberattacks [2], [3], including Denial of service (DoS) attacks that disable target systems by flooding them with huge streams of requests [4]. While many such attacks go unreported when they occur, just one Distributed DoS attack in 2017 targeting Google, compromised $180,000$ web servers which flooded Google servers at overall bitrates of 2.54 Tera-bits/sec [5]. Other attacks aim mainly at the IoT [6]–[8], while Botnet attacks [9] are particularly vicious since they spread by inducing their victims to become attackers [10], [11]. UDP Flood attacks are also exploited by Botnets to create massive congestion that overwhelms network nodes and ports. Using spoofed-source-address UDP packets, they cause their victims to crash due to high traffic volumes, creating a denial of service, causing lost data and resulting in missing and incomplete readings of the data carried by legitimate IoT traffic [12], [13]. Such attacks can be particularly dangerous for cyber-physical systems such as Supply Chains where complex timing interactions are threatened by cyberattacks, leading to costly loss and delayed delivery of products and service. Thus securing cyber-physical systems [14]–[16] has been of growing concern, especially in Supply Chains [17], autonomous vehicles [18], and systems where different types of IoT devices with different datasets must concurrently operate [19], [20] in a common physical and virtual framework [21].

A. Prior Work

The concern about cybersecurity has motivated a large literature on cyberattacks, and about Attack Detection (AD) methods [22], installed at servers and edge devices that support IoT systems, rather than in the Cloud. Typically evaluated for accuracy using statistics [8], [23], [24], such learning based [25] AD algorithms are often tested under ideal conditions on general purpose computers [26]–[30], with attack traffic treated as data, where the attack's overload on the performance of the victim node is not measured.

Various AD test-beds [31] for cyber-physical and IoT networks are presented in [32]–[34]. Experiments on wind-farms under SYN attacks are discussed in [35] and other experimental IoT security studies can be found in [36], [37]. Data collection and display for flood attacks are discussed in [38], while in [39] real-time data collection for IoT DNS attacks is presented. Denial of Service (DoS) attacks against Software Defined Networks for the IoT was studied in [40], also concerns attack emulation without the overload caused by attacks against entities such as autonomous vehicles [41] and IoT servers [42].

B. Motivation and Research Plan

The present paper is motivated by the need to:
- Experimentally evaluate the effect of IoT Server overload during an ongoing UDP Flood attack, and understand the attack's impact on the Server's capacity to carry out Attack Detection (AD) and other useful processing functions,
- Demonstrate a system architecture, and a traffic shaping policy [43] that was initially proposed to mitigate the IoT's Massive Access Problem (MAP) [44]–[46], to guarantee that in the presence of attacks that create large packet flows, the Server can operate seamlessly and accomplish its role for AD and other useful IoT processing functions,
- Experimentally show that mitigation actions can be triggered

to rapidly eliminate the long-term effects of such UDP Flood attacks from the system as a whole.

Thus in Section II, we provide new measurements on the experimental test-bed shown in Figure 1, to illustrate the effect of a UDP Flood attack emanating from an IoT traffic source against the IoT Server that receives packets from different IoT devices. These measurements show that the Server is significantly impacted during an attack and is impeded from conducting its AD functions in a timely fashion.

Based on this observation, Section III proposes and evaluates a novel system architecture shown in Figure 6 where the Server is preceded by a Smart "Quasi-Deterministic Forwarding Policy (QDP)" Forwarder (SQF) that shapes the traffic that is forwarded to the Server. Our results show that if we select the SQF parameters based on mathematical principles [43], then the SQF effectively limits the undesirable effects of an attack against the Server. However, attack packets accumulate at the SQF which protects the Server, and mitigation actions may discard the accumulated attack traffic.

II. Initial Experiments

We first conducted experiments on the Local Area Network (LAN) test-bed for a system shown in Figure 1, in which IoT devices represented by several Raspberry Pi machines, send UDP traffic to the Server. One of the Pi machines is also programmed to generate attack traffic either at predetermined instants or at random. These Raspberry Pi 4 Model B Rev 1.2 machines (RPi1 and RPi2) machines, each have a 1.5GHz ARM Cortex-A72 quad-core processor and 2GB LPDDR4 $- 3200$ SDRAM, running Raspbian GNU/Linux 11 (bullseye), a Debian-based operating system optimized for Raspberry Pi hardware. The normally operating (uncompromised) Raspberry PIs periodically send UDP Protocol packets to the Server, containing their own temperature measurements, to the Server shown in Figure 1. This choice of data is an example of real data available to the Raspberry Pis.

The Server is an Intel 8-Core i7 $- 8705$G with 16GB of RAM, a 500GB hard drive, and a Linux $5.15.0 - 60-$ generic $66-$Ubuntu SMP operating system. It runs at 3.1Ghz and communicates with the Raspberry Pis via Ethernet Local Area Network (LAN), and receives IoT traffic from them via UDP. Figure 2 shows that the Server supports the UDP protocol with SNMP for incoming packets, and operates the accurate AD algorithm in [30], using the Random Neural Network [47] and its extension [48]. The UDP protocol's simplicity fits the needs of the simple IoT devices that we use, since UDP does not establish a connection before transmitting and does not use ACKs or error recovery for communications [49].

While many datasets are used to generate attack traffic, including the KDD99 or its improved version NSL-KDD, UNSW-NB15, CICDS2017, and the Bot-IoT dataset [39], in this work we use MHDDoS [50] containing 56 recent real-world DoS attacks with 56 different techniques for attack traffic emulation.

In Figure 3 we show measurements of the effect of a 60 sec Flood Attack, which overwhelms the 8-Core Server with

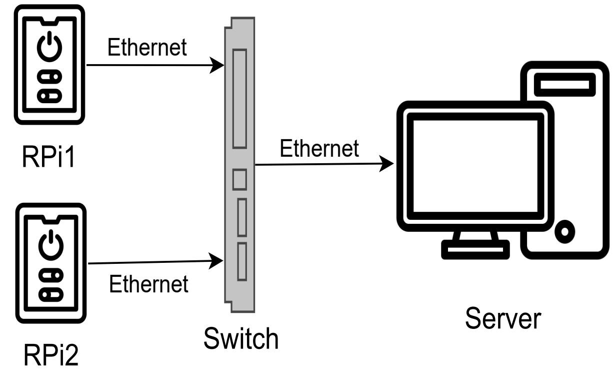

Fig. 1. The test-bed composed of an Ethernet network with Raspberry Pi machines that generate normal traffic, as well as possible traffic. An IoT Server receives the IoT traffic via the same network.

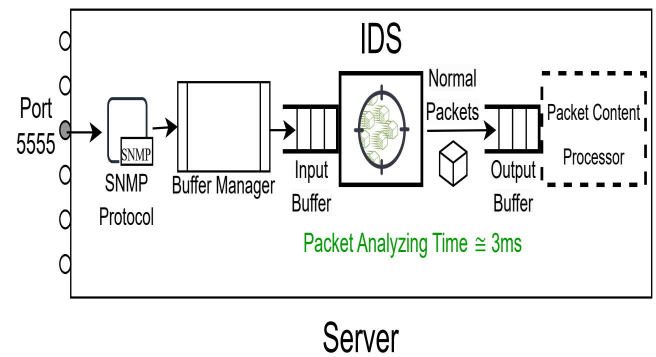

Fig. 2. Internal software architecture of the Server, containing a SNMP network protocol manager, an AD system [30] that identifies attack packets, and software for processing the incoming packet contents.

some $400,000$ packets that accumulate at its input buffer. The Server's activities, including AD, are paralyzed by the attack and the packet backlog takes nearly 300 minutes to clear. Thus, we see that the attack significantly impairs The Server's capability for AD, its ability to discard attack packets and even to process benign packets.

The detailed measurements of the Server's AD processing times per packet, when there is no attack, and when a UDP Flood Attack occurs, are reported in Figure 4 and Figure 5.

We observe that the Server's AD processing time per packet, when **no attacks** occur, has an average value of 2.98 milliseconds (ms). On the other hand, when the Server is targeted by a UDP Flood Attack, we observe a substantial increase in the AD algorithm's average processing time to 4.82 ms. Moreover, the AD processing time per packet when the Server is under attack, exhibits some very large outliers, as shown in Figure 5. We observe that these "outlier" processing times are close to 10^3 times larger than the typical values, showing that during a UDP Flood Attack the Server's AD procesing of packets is repeatedly paralyzed for a substantial amount of time, as also shown in Figure 3.

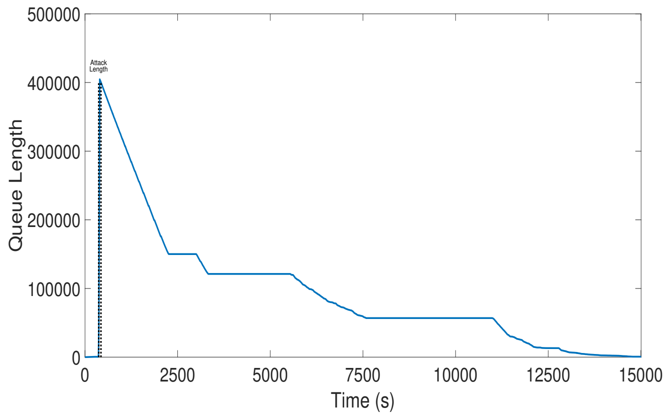

Fig. 3. Experimentally measured queue length (the y-axis is in number of packets) over time (the x-axis is in seconds) at the Server input prior to processing at the AD module, during a 60 second UDP Flood Attack launched from one of the Rasberry Pis of Figure 1 against the Server. The backlog of packets at the Server initially rises rapidly to $400,000$ packets, and without human intervention the congestion at the Server lasts far longer than the attack itself, up to several hours, due to the fact that the Server is paralyzed and stops its AD processing packets for long time intervals. These long interruptions in AD processing time are observed as the large outliers in AD processing times in Figure 5.

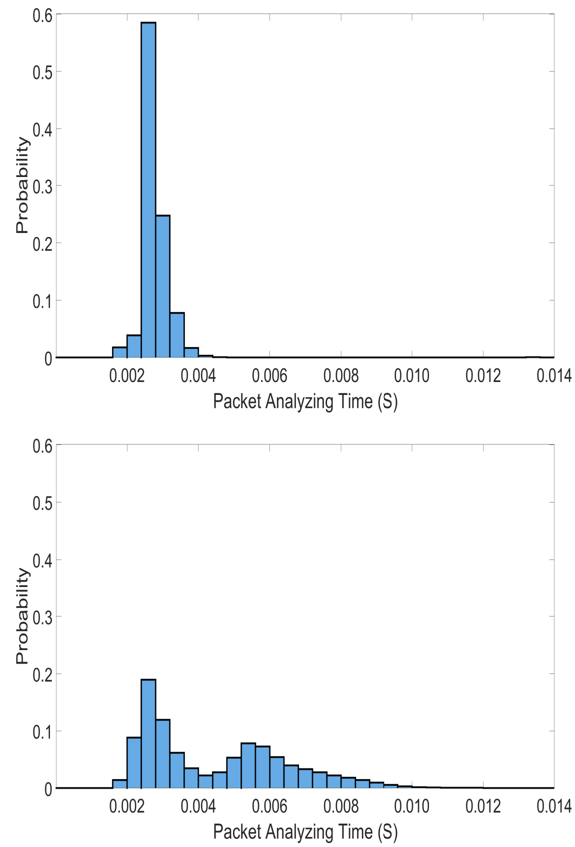

Fig. 4. In the Upper figure, we show the histogram of measurements of the Server's AD processing time per packet, when there is no attack, exhibiting an average processing time of 2.98 ms and variance 0.0055 ms^2. In the Lower figure an attack is occurring: the Server's measured average AD processing time of packets rises substantially to 4.82 ms with a variance 0.51 ms^2,

A. Lindley's Equation when the SQF is not Used

If the SQF module is **not** being used as shown in Figure 1:

- Let $0 = a_0 \leq a_1 \leq a_2, \ ...$, be the successive packet arrival instants at the Server through the Ethernet LAN from any of the IoT devices connected to the LAN. We also define the interarrival time $A_{n+1} = a_{n+1} - a_n$.
- Let T_n denote the Server's AD processing time for the $n-th$ packet, and assume that the Server processes packets in First Come First Served (FCFS) order.

Then the total waiting time L_{n+1} of the $n+1$-th incoming packet, between the instant a_n and the start of the AD processing time of the Server, is given by the well known Lindley's equation:

$$L_{n+1} = [L_n + T_n - A_{n+1}]^+, \ n \geq 0, \ L_0 = 0 \ , \qquad (1)$$

where for a real number X, we use the notation:

$$[X]^+ = X \ if \ X > 0, \ and \ [X]^+ = 0 \ if \ X \leq 0. \qquad (2)$$

Note that $L_0 = 0$ because the first incoming packet encounters an empty queue infront of the AD. Note also that whenever we have $T_n > A_{n+1}$ then $L_{n+1} > L_n$, i.e. the waiting time increases.

During a Flood Attack, the values of A_n and T_n will be modified, as we see from Figure 3, indicating that packet arrival rates have considerably increased so that the values of A_n are much smaller, while Figure 4 shows that the values of T_n are also larger. However the form of (1) does not change.

III. The QDTP based Forwarding Element

In Figure 6, we present our proposed modified architecture where the Server, whose role is to process incoming IoT packets – including operating the AD module in order to detect attacks – is "protected" by a Smart QDTP Forwarder (SQF) which is placed between the Ethernet based sources of IoT traffic, and the Server's input port. The SDF's role is to shape the incoming traffic directed at the Server using the *Quasi-Deterministic Transmission Policy (QDTP)* [43], [51], [52].

QDTP is a simple policy that delays some of the packets it receives, by forwarding them to the Server at time $t_n \geq a_n$, where a_n is the n-th packet's arrival instant to the SQF, and t_n is the instant at which SQF forwards the packet to the Server, and is defined by:

$$t_{n+1} = \max\{t_n + D, a_{n+1}\}, \ t_0 = a_0, \ n \geq 0, \qquad (3)$$
$$so \ that \ t_{n+1} - t_n \geq D \ , \qquad (4)$$

where $D > 0$ is a constant parameter of the QDTP algorithm that needs to be fixed. When the $n-th$ packet is transmitted by the SQF, we assume that it arrives instantaneously at the Server's input queue for AD processing. Here we are in fact assuming that the physical transmission time from the SQF to the Server, and the network protocol service time inside the Server, are tiny compared to the AD processing duration T_n at the Server. Thus the total delay Q_n experienced by the n-

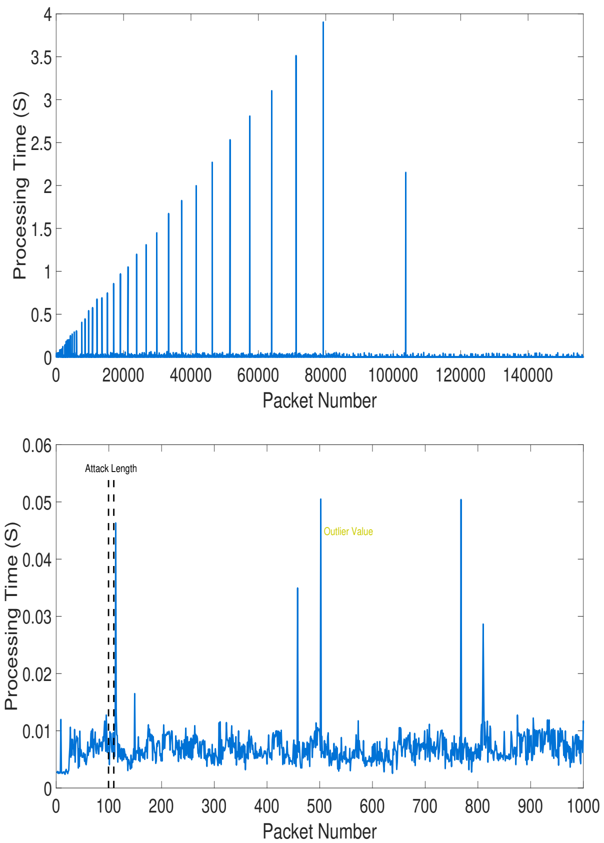

Fig. 5. In the Upper figure, we show successive measurements of the Server's AD processing time per packet during a UDP Flood Attack (in the absence of the QDTP Forwarder SQF), showing large outliers that initially become more severe, and gradually become less frequent over time. In the Lower figure, the AD processing time of packets that is measured after the UDP Attack begins, reveals very large outliers in AD processing times, indicating that the AD is intermittently paralyzed or unable to operate.

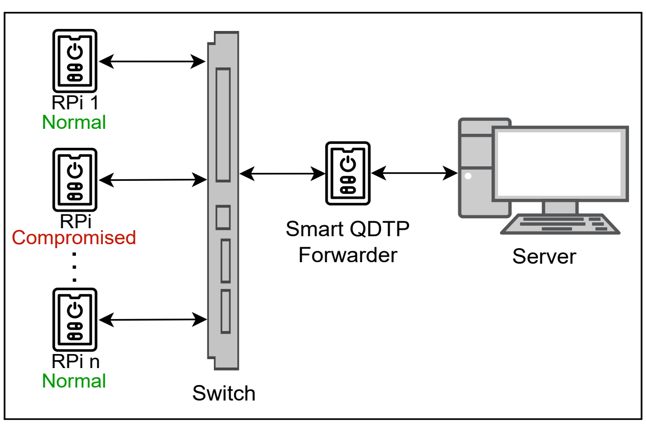

Fig. 6. The figure shows the modified system architecture where a Smart QDTP Forwarder (SQF) constantly acts as a traffic shaping interface between the Ethernet LAN and the Server. The effect of the SQF is to eliminate the paralyzing effect of the packet flood at the Server, buffering packets within the SQF and forwarding in a manner which allows the Server to conduct its AD processing and other work in a timely fashion.

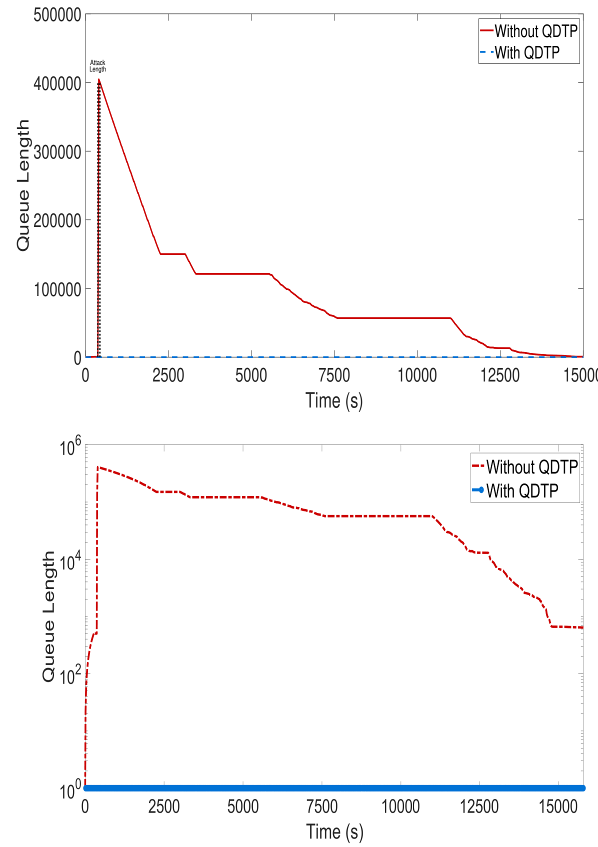

Fig. 7. The queue length at the Server in the presence of a 60 second UDP Flood Attack. The figure Above shows the case **without** the SQF, and we see that the queue length peaks to $400,000$ packets and descends slowly over some $15,000$ seconds. The figure Below compares the queue length in logarithmic scale, **with SQF in Blue** using the parameter $D = 3\ ms$, against the case **without SQF in Red**, with the same UDP Flood Attack which lasts 60 seconds. Note that because the value of D we use is very close to the average value of T_n measured to be 2.98 ms in the absence of an attack, as shown in Figure , the fluctuations in the values of T_n will cause a small queue buildup (in the order of a few packets), as seen in te Blue plot in the figure Below.

th packet due to the action of the SQF, that elapses from the arrival of the n-th packet to the SQF at a_n, until its arrival to the AD at the Server at t_n, is:

$$Q_0 = t_0 - a_0 = 0, \quad (5)$$

$$\begin{aligned} Q_{n+1} &= t_{n+1} - a_{n+1}, \\ &= \max\{t_n + D, a_{n+1}\} - a_{n+1}, \\ &= 0,\ if\ t_n + D \leq a_{n+1},\ and \\ &= t_n + D - a_{n+1},\ otherwise. \end{aligned} \quad (6)$$

Since $t_n = Q_n + a_n$, we obtain the recursive expression:

$$\begin{aligned} Q_{n+1} &= [t_n + D - a_{n+1}]^+, \\ &= [Q_n + D - A_{n+1}]^+,\ n \geq 0, \end{aligned} \quad (7)$$

which is also an instance of Lindley's equation (1).

On the other hand, the Server's AD module also acts as a FCFS queue and we can exploit Lindley's equation again

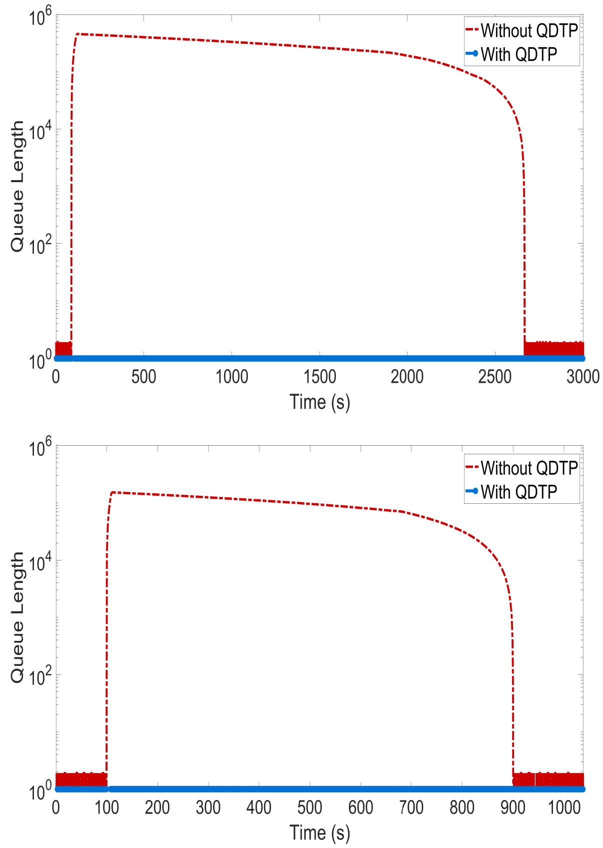

Fig. 8. We measure the Server queue length (repesented logarithmically) when the Server is targeted by a UDP Flood Attack that lasts $30 - sec$ (Above) and $10 - sec$ (Below). The Red curves show the case without the SQF traffic shaping Forwarder, while the Blue Curves show the effect of the use of the SQF which uses QDTP. We observe the huge difference in queue length. For both the 30 and 10 second attacks, we have set $D = 3\ ms$.

to compute $W_n,\ n \geq 0$ the waiting time of the n-th packet that arrives to the Server to be processed for attack detection, which is:

$$
\begin{aligned}
W_{n+1} &= [W_n + T_n - (t_{n+1} - t_n)]^+,\ W_0 = 0, && (8)\\
&\leq W_n + T_n - (t_n - t_{n+1}), && (9)
\end{aligned}
$$

since the n-th packet's AD service time is T_n and the $n+1$-th interarrival interval to the Server's AD queue is $t_{n+1} - t_n$.

Therefore using equations (9) and (4) we obtain:

$$W_{n+1} \leq W_n + T_n - D, \qquad (10)$$

and we have the following key insight into how to choose D:

Result 1. If we fix the parameter D in the QDTP policy for the SQF to a value so that $D > T_n$ for all $n \geq 0$, then the waiting time W_n at the Server will remain at the value $W_n = 0$ for all $n \geq 0$.

We now present experiments showing the usefulness of Result 1. Noting from Figure 4 that the measured average value of T_n is 2.98 ms when there is no attack, we first select $D = 3\ ms$ which is just above that value.

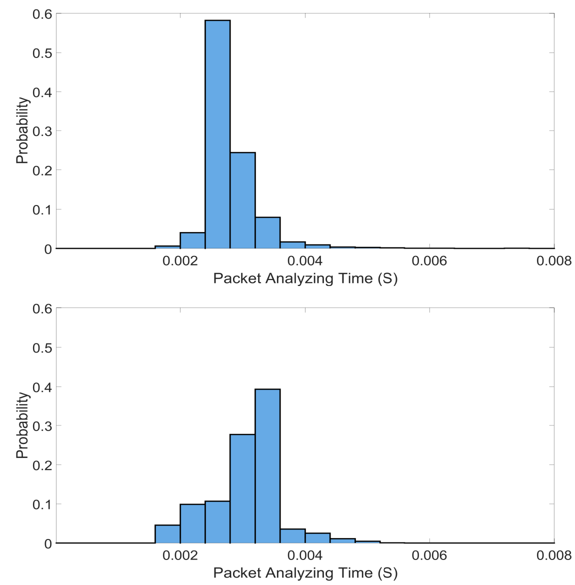

Fig. 9. AD Processing Time at the Server when the SQF with the QDTP Policy is installed and the parameter $D = 2.7\ ms$ is used. We observe that the AD processing time T_n has an average value of 2.97 ms and variance of 0.0041 sec^2 in the absence of an attack (Above). In the presence of a UDP Flood Attack (Below) the average processing time of the AD per packet is higher by roughly 10% on average, at 3.28 ms with a variance of 0.0023 sec^2 so that the SQF is effective in protecting the Server from paralysis and excessive slowdown.

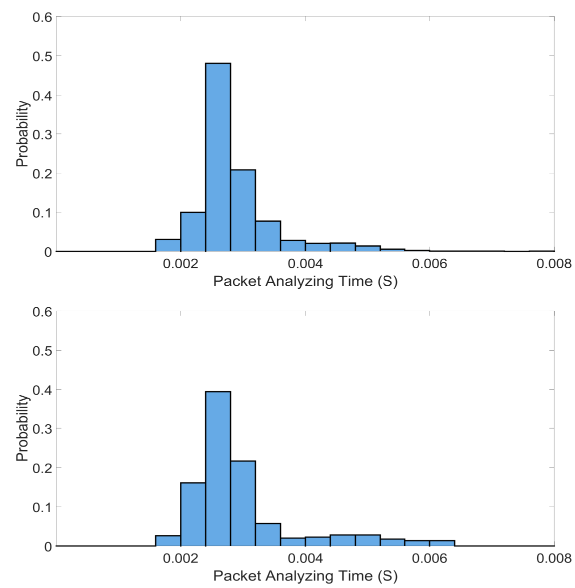

Fig. 10. AD Processing Time at the Server when the SQF with the QDTP Policy is installed and the parameter $D = 3.20\ ms$ is used. We observe that the AD processing time T_n has an average value of 3.00 ms and variance of 0.0036 sec^2 in the absence of an attack (Above). In the presence of a UDP Flood Attack (Below) the average processing time of the AD per packet is quasi-identical on average, at 2.99 ms with a variance of 0.0067 sec^2 so that in this case too, the SQF is effective in protecting the Server from paralysis and excessive slowdowns.

Figure 7 compares the case **without SQF** (Above) and **with SQF** (Below) during a 60 sec UDP Flood Attack. Note that the

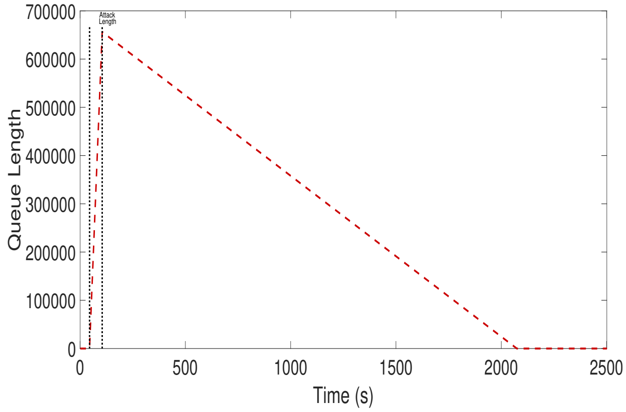

Fig. 11. SQF queue length (y-axis in number of packets) against time (x-axis in seconds) when a UDP Flood attack lasts for 60 seconds. We have used $D = 3\ ms$, and no mitigation action takes place.

figure Above represents the Server queue length varying over time, without the SQF. The figure Below is in logarithmic scale for the Server queue length, and compares the cases without SQF (in Red) and with SQF (in Blue) for the Server queue length varying over time. Since $D = 3\ ms$ is very close to the average of T_n, the fluctuations in the values of T_n cause a small queue buildup of a few packets, as seen in the Blue plot in the lower part of the shown Below.

Figure 8 shows the results of four experiments where we measure the queue length at the Server when a UDP Flood Attack lasts 30 (Above) and 10 (Below) seconds, without (Red) and with (Blue) the SQF Forwarder. Without SQF, the Server's AD processing time increases significantly. In the 30 sec attack, approximately $470,000$ packets are received at the Server and without SQF it takes 44.45 minutes for the Server to return to normal process them, while in the 10 sec attack $153,667$ packets are received and it takes the Server roughly 15 minutes to process them. Note that in these curves it takes some 99 seconds for the compromised RPi to launch the attack.

Figure 9 shows that when we use the SQF based system with $D = 2.7\ ms$, which is smaller than the value recommended by Result 1, when there is no attack this choice of D has very little effect. However, when a UDP Flood Attack occurs, the Server's AD processing is somewhat slowed down and the average value of T_n increases by roughly 10%.

On the other hand, Figure 10 confirms **Result 1** since it shows that, if we take $D = 3.2\ ms$ which guarantees that $D > T_n$ most of the time, then the measured average value of T_n remains at around 3 ms showing that it has not been slowed down by the attack's overload effect. Of course the same is seen when no attack occurs.

IV. Adaptive Algorithm for Mitigating Attacks

When a Flood Attack occurs, the SQF accumulates packets in its input queue, and forwards them to the Server using the QDTP algorithm with $D = 3\ ms$, so that the Server does not experience any AD slowdown, ensuring that the Server continues to operate as usual. Figure 11 shows the sudden increase and then slow decrease of the SQF input queue when a UDP Flood attack lasts for 60 seconds, and the SQF uses $D = 3\ ms$. Since both the SQF (and the Server) do not drop packets, the attack packets will accumulate at the input queue of the SQF.

Thus in this section we test a possible mitigating action that the SQF can take. Since Flood Attacks are characterized by an unusually high packet arrival rate, and this is also one of the attack detection parameters used by the AD used in this work [30], we now test an additional feature, as follows, for the parameters N and K of the mitigating action:

1) If the SQF receives more than N packets in a time interval smaller than or equal to D, it drops all incoming packets for the next $K.D$ time units.
2) The action is repeated as long as the condition 1) (above) on N persists.

To illustrate the effect of this simple policy, we first set $N = 10$ and $K = 3$ and implement the proposed drop-based mitigation policy. In this experiment, an RPi launches a 10 second Flood Attack, and the resulting queue length at the input of the SQF is shown in Figure 12, where we see that the SQF input buffer reaches a small value of 12 packets. The attack starts at the 34-th second and lasts 10 seconds, but thanks to the mitigation policy there is no accumulation of packets. After the attack ends the SQF can continue to operate normally.

Figure 13 displays the queue length of the SQF input buffer in a second experiment, when the attack lasts 60 seconds, showing similar results to the first experiment. Both measurements show the importance of having a simple mitigating action to deal with high volume Flood Attacks. However, although this policy appears attractive, it comes at the cost of dropping legitimate (non-attack) packets that come from non-compromised IoT devices.

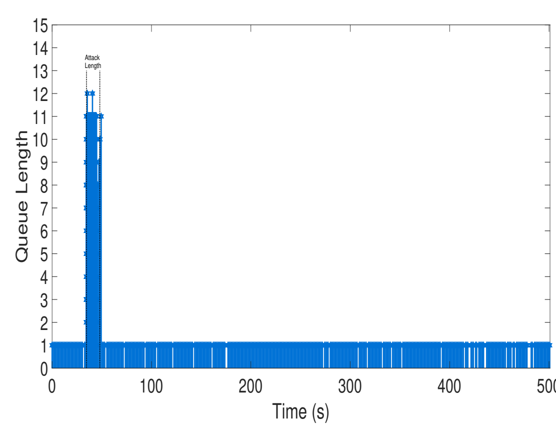

Fig. 12. SQF Smart QDTP Forwarder queue length when the attack lasts for 10 seconds, with the mitigation action, and $D =\ 3ms$.

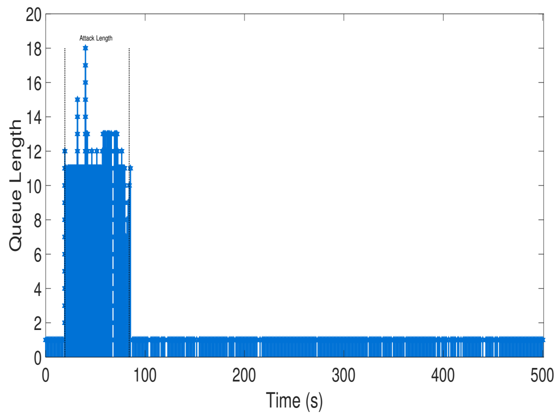

Fig. 13. SQF queue length with $D = 3\ ms$ when the attack lasts for 60 seconds, and the mitigation action is also applied.

V. Conclusions

This paper has considered the effect of UDP Flood attacks on an IoT Server that processes incoming traffic from IoT devices via a local area network. The Server also incorporates an AD module, and we first show that such attacks, even when they last just a few seconds, create overload for the IoT Server. As a resut, its normal operations, including AD, are substantially slowed down, and a 60 second attack may create a backlog of packets at the Server that requires several hours to clear out. ma

Thus we propose that the Server's input be "protected" by a special SQF front-end that operates the QDTP policy, to allow the timely operation of the Server even when an attack occurs. This approach requires that an inexpensive lightweight hardware addition, such as an RPi, be installed between the local area network that supports the IoT devices and the Server. Several experiments are used to illustrate the effectiveness of the proposed approach. However, the SQF with its QDTP policy requires that a key timing parameter D be chosen. Therefore, we provide a theoretical analysis of how D should be selected: we show that it must be just larger than the AD processing time of the Server under normal, i.e. non-attack, conditions. We then validate this observation with several experiments and show that the SQF can preserve the Server from congestion and overload, and allow it to operate normally. However, we note that the congestion that has been eliminated at the Server may now accumulate at the SQF input, although this in itself does not stop the RPi based SQF from continuing its normal operation.

Furthermore, when the incoming traffic rate is such that it clearly indicates an attack, or when the Server informs the SQF that an attack is occuring, we can implement a mitigating action at the SQF to drop incoming packets in relatively short successive time intervals. This approach is tested experimentally and shown to be effective. However, the fact that such a policy may also drop incoming legitimate packets implies that there will be circumstances when it cannot be used and a close coupling between AD at the Server and packet drop actions at the SQF will be needed.

While this paper has focused on an architecture with multiple sources of IoT traffic represented by several RPis, future work will consider Edge Systems having multiple IoT servers, as well as multiple IoT devices and packet sources, and will study the usage of dynamic policies for AD and traffic routing at the edge for complex IoT Server and SQF gateway architectures that can be used in cyber-physical systems such as Supply Chains. Another important issue that should be addressed is the energy consumption of such edge systems [53], [54], so that dynamic management policies may be used to minimize energy consumption, optimize Quality of Service and Cybersecurity.

Acknowledgements

Research supported by the European Commission Horizon Europe Programme for Research and Innovation (2021-2027), DOSS Project under Grant Agreement No: 101120270.

References

[1] Cisco, *Cisco Annual Internet Report (2018–2023)*, Mar. 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[2] J. Liu, L. Song *et al.*, "A novel congestion reduction scheme for massive machine-to-machine communication," *IEEE Access*, vol. 5, pp. 18 765–18 777, 2017.

[3] L. Tello-Oquendo, I. Leyva-Mayorga, V. Pla, J. Martinez-Bauset, J.-R. Vidal, V. Casares-Giner, and L. Guijarro, "Performance analysis and optimal access class barring parameter configuration in LTE-A networks with massive M2M traffic," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 4, pp. 3505–3520, 2018.

[4] E. Johns and M. Ell, "Cyber security breaches survey 2023," April 2023. [Online]. Available: https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2023/cyber-security-breaches-survey-2023

[5] Cloudflare. [Online]. Available: https://www.cloudflare.com/learning/ddos/famous-ddos-attacks/

[6] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan, "Internet of Things (IoT) security: Current status, challenges and prospective measures," in *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE, 2015, pp. 336–341.

[7] M. U. Farooq, M. Waseem, A. Khairi, and S. Mazhar, "A critical analysis on the security concerns of Internet of Things (IoT)," *International Journal of Computer Applications*, vol. 111, no. 7, pp. 1–6, 2015.

[8] S. Evmorfos, G. Vlachodimitropoulos, N. Bakalos, and E. Gelenbe, "Neural network architectures for the detection of SYN flood attacks in IoT systems," in *Proceedings of the 13th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, 2020, pp. 1–4.

[9] N. Statt, "How an army of vulnerable gadgets took down the web today," October 2016. [Online]. Available: https://www.theverge.com/2016/10/21/13362354/dyn-dns-ddos-attack-cause-outage-status-explained

[10] H. Sinanović and S. Mrdovic, "Analysis of mirai malicious software," in *2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. IEEE, 2017, pp. 1–5.

[11] M. Antonakakis, et al., "Understanding the Mirai Botnet," in *Proceedings of the 26th USENIX Security Symposium*, 2017. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis

[12] A. Iqbal, S. Aftab, I. Ullah, M. A. Saeed, and A. Husen, "A classification framework to detect DoS attacks." *International Journal of Computer Network & Information Security*, vol. 11, no. 9, 2019.

[13] T. Chin, X. Mountrouidou, X. Li, and K. Xiong, "Selective packet inspection to detect DoS flooding using software defined networking (sdn)," in *2015 IEEE 35th International Conference on Distributed Computing Systems Workshops*, 2015, pp. 95–99.

[14] A. Levi, E. Çetintaş, M. Aydos, Ç. K. Koç, and M. U. Çağlayan, "Relay attacks on bluetooth authentication and solutions," in *2004 International Symposium on Computer and Information Sciences, LNCS 3280*. Springer Cham, 2004, pp. 278–288.

[15] M. Akgün and M. U. Çağlayan, "Providing destructive privacy and scalability in RFID systems using PUFs," *Ad Hoc Networks*, vol. 32, pp. 32–42, 2015.

[16] M. U. Çağayan, "Review of some recent European Cybersecurity Research and Innovation Projects," *Infocommunications Journal*, vol. 14, no. 4, pp. 70–78, 2022. [Online]. Available: https://doi.org/10.36244/ICJ.2022.4.10

[17] N. M. Gökhan, "Development of a simultaneous design for supply chain process for the optimization of the product design and supply chain configuration problem," *Engineering Management Journal*, vol. 2, no. 4, p. 20–30, 2010.

[18] A. Frötscher, B. Monschiebl, A. Drosou, E. Gelenbe, M. J. Reed, and M. Al-Naday, "Improve cybersecurity of c-its road side infrastructure installations: the seriot-secure and safe iot approach," in *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*. IEEE, 2019, pp. 1–5.

[19] S. R. Pokhrel and J. Choi, "Federated learning with blockchain for autonomous vehicles: Analysis and design challenges," *EEE Transactions on Communications*, vol. 68, no. 8, p. 4734–4746, 2020.

[20] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive iot networks," *IEEE Internet of Things Journal*, no. 5, p. 4641–4654, 2020.

[21] Z. Xu, F. Yu, J. Xiong, and X. Chen, "Helios: Heterogeneity-aware federated learning with dynamically balanced collaboration," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 2021, p. 997–1002.

[22] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: classification and state-of-the-art," *Computer networks*, vol. 44, no. 5, pp. 643–666, 2004.

[23] G. Oke, G. Loukas, and E. Gelenbe, "Detecting denial of service attacks with bayesian classifiers and the random neural network," in *2007 IEEE International Fuzzy Systems Conference*, 2007, pp. 1–6.

[24] H. Ahmetoglu and R. Das, "A comprehensive review on detection of cyber-attacks: Data sets, methods, challenges, and future research directions," *Internet of Things*, vol. 20, p. 100615, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S254266052200097X

[25] E. Gelenbe, "Learning in the recurrent random neural network," *Neural Computation*, vol. 5, no. 1, pp. 154–164, 1993.

[26] M. Banerjee and S. Samantaray, "Network traffic analysis based iot botnet detection using honeynet data applying classification techniques," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 17, no. 8, 2019.

[27] T. A. Tuan, H. V. Long, R. Kumar, I. Priyadarshini, N. T. K. Son *et al.*, "Performance evaluation of Botnet DDoS attack detection using machine learning," *Evolutionary Intelligence*, pp. 1–12, 2019.

[28] A. I. Al-issa, M. Al-Akhras, M. S. ALsahli, and M. Alawairdhi, "Using machine learning to detect DoS attacks in wireless sensor networks," in *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, 2019, pp. 107–112.

[29] E. Y. Güven and Z. Gürkaş-Aydın, "Mirai botnet attack detection in low-scale network traffic," *Intelligent Automation & Soft Computing*, vol. 37, no. 1, pp. 419–437, 2023.

[30] E. Gelenbe and M. Nakıp, "Traffic based sequential learning during botnet attacks to identify compromised IoT devices," *IEEE Access*, vol. 10, pp. 126 536–126 549, 2022.

[31] J. Mirkovic, S. Fahmy, P. Reiher, and R. K. Thomas, "How to test dos defenses," in *2009 cybersecurity applications & technology conference for homeland security*. IEEE, 2009, pp. 103–117.

[32] M. Kaouk, F.-X. Morgand, and J.-M. Flaus, "A testbed for cybersecurity assessment of industrial and IoT-based control systems," in *Lambda Mu 2018 - 21è Congrè de Maîtrise des Risques et Sûreté de Fonctionnement, Oct 2018, Reims, France*. [Online]. Available: https://hal.science/hal-02074654v1/document

[33] M. Annor-Asante and B. Pranggono, "Development of smart grid testbed with low-cost hardware and software for cybersecurity research and education," *Wireless Personal Communications*, vol. 101, pp. 1357–1377, 2018.

[34] O. A. Waraga, M. Bettayeb, Q. Nasir, and M. A. Talib, "Design and implementation of automated iot security testbed," *Computers & security*, vol. 88, p. 101648, 2020.

[35] V. K. Singh, R. Sharma, and M. Govindarasu, "Testbed-based performance evaluation of attack resilient control for wind farm scada system," in *2020 IEEE Power & Energy Society General Meeting (PESGM)*. IEEE, 2020, pp. 1–5.

[36] A. Ghaleb, S. Zhioua, and A. Almulhem, "Scada-sst: a scada security testbed," in *2016 World Congress on Industrial Control Systems Security (WCICSS)*. IEEE, 2016, pp. 1–6.

[37] A. Tesfahun and D. L. Bhaskari, "A scada testbed for investigating cyber security vulnerabilities in critical infrastructures," *Automatic Control and Computer Sciences*, vol. 50, pp. 54–62, 2016.

[38] S.-U. Park and S.-M. Hwang, "Test bed construction for apt attack detection," *International Journal of Control and Automation*, vol. 11, no. 4, pp. 175–186, 2018.

[39] R. Arthi and S. Krishnaveni, "Design and development of iot testbed with ddos attack for cyber security research," in *2021 3rd International Conference on Signal Processing and Communication (ICPSC)*. IEEE, 2021, pp. 586–590.

[40] A. P. Wright and N. Ghani, "A testbed for the evaluation of denial of service attacks in software-defined networks," in *2019 SoutheastCon*. IEEE, 2019, pp. 1–6.

[41] P. V. Sontakke and N. B. Chopade, "Impact and analysis of denial-of-service attack on an autonomous vehicle test bed setup," in *Proceedings of Third International Conference on Intelligent Computing, Information and Control Systems: ICICCS 2021*. Springer, 2022, pp. 221–236.

[42] M. Nasereddin, M. Nakıp, and E. Gelenbe, "Measurement based evaluation and mitigation of flood attacks on a lan test-bed," in *The 48th IEEE Conference on Local Computer Networks (LCN) October 1-5, 2023, Daytona Beach, Florida, USA*. IEEEXpress, 2023, pp. 1–4. [Online]. Available: https://zenodo.org/record/8094796

[43] E. Gelenbe and K. Sigman, "Iot traffic shaping and the massive access problem," in *ICC 2022, IEEE International Conference on Communications, 16–20 May 2022, Seoul, South Korea*, no. https://zenodo.org/record/5918301. https://zenodo.org/record/5918301, 2022, pp. 1–6.

[44] V. Rodoplu, M. Nakıp, D. T. Eliiyi, and C. Güzelis, "A multi-scale algorithm for joint forecasting-scheduling to solve the massive access problem of IoT," *IEEE Internet of Things Journal*, 2020.

[45] V. Rodoplu, M. Nakip, R. Qorbanian, and D. T. Eliiyi, "Multi-channel joint forecasting-scheduling for the Internet of Things," *IEEE Access*, vol. 8, pp. 217 324–217 354, 2020.

[46] M. Nakip and E. Gelenbe, "Randomization of data generation times improves performance of predictive IoT networks," in *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*, 2021, pp. 350–355.

[47] E. Gelenbe, "Random neural networks with negative and positive signals and product form solution," *Neural computation*, vol. 1, no. 4, pp. 502–510, 1989.

[48] E. Gelenbe and Y. Yin, "Deep learning with dense random neural networks," in *International Conference on Man–Machine Interactions*, no. DOI:10.1007/978-3-319-67792-. Springer, Cham, 2017, pp. 3–18.

[49] S. Kumar and S. Rai, "Survey on transport layer protocols: Tcp & udp," *International Journal of Computer Applications*, vol. 46, no. 7, pp. 20–25, 2012.

[50] "MHDDoS - DDoS Attack Script With 56 Methods," Online, May 2022, accessed: 2023-02-22. [Online]. Available: https://github.com/MatrixTM/MHDDoS

[51] E. Gelenbe, M. Nakip, D. Marek, and T. Czachorski, "Diffusion analysis improves scalability of IoT networks to mitigate the massive access problem," pp. 182–189, November 2021.

[52] J. Bergquist, E. Gelenbe, and K. Sigman, "On an Adaptive-Quasi-Deterministic Transmission Policy queueing model," *Stochastic Systems, submitted for publication, month = June, year = 2023*.

[53] E. Gelenbe, "Energy packet networks: ICT based energy allocation and storage," in *International Conference on Green Communications and Networking*. Springer, 2011, pp. 186–195.

[54] P. Fröhlich, E. Gelenbe, J. Fiołka, J. Checinski, M. Nowak, and Z. Filus, "Smart sdn management of fog services to optimize QoS and energy," *Sensors*, vol. 21, no. https://doi.org/10.3390/s21093105, p. 3105, 2021.