Proposal of CYPHONIC end-device functions on Windows OS

Guilherme Seidy Ujiie
Aichi Institute of Technology,
Toyota, Aichi 470-0392, Japan
Useidy0201@pluslab.org

Ren Goto
Aichi Institute of Technology,
Toyota, Aichi 470-0392, Japan
r0719en@pluslab.org

Kazushige Matama
Aichi Institute of Technology,
Toyota, Aichi 470-0392, Japan
matama@pluslab.org

Yoshiya Suzuki
Aichi Institute of Technology,
Toyota, Aichi 470-0392, Japan
homare448@pluslab.org

Hidekazu Suzuki
Meijo University,
Nagoya, Aichi 468-0073, Japan
hsuzuki@meijo-u.ac.jp

Katsuhiro Naito
Aichi Institute of Technology,
Toyota, Aichi 470-0392, Japan
naito@pluslab.org

2023 IEEE 12th Global Conference on Consumer Electronics (GCCE) | 979-8-3503-4018-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/GCCE59613.2023.10315250

***Abstract*—The authors are currently engaged in researching and developing CYber PHysical Overlay Network over Internet Communication (CYPHONIC), a secure communication technology to achieve communication connectivity and mobility transparency. The conventional version of CYPHONIC is primarily developed for Linux terminals, and as of now, the terminal functionality for other operating systems remains unimplemented. Considering the widespread adoption of Windows in the market, implementing terminal functionality for Windows is considered essential for future development. In this paper, we focus on designing and implementing the terminal functionality of CYPHONIC, specifically for Windows. In our approach, we use a virtual interface using Windows-specific TUN/TAP and proceed to design and implement the terminal functionality, which leverages Windows network socket, thus enabling the utilization of CYPHONIC nodes on Windows platform. Through performance evaluations conducted using a prototype implementation, we have confirmed that it is feasible to achieve communication performance equivalent to the terminal functionality available on Linux.**

***Index Terms*—NAPT traversal, Mobility, Zero-trust network, Overlay network, Windows**

I. Introduction

Internet communication serves various purposes, and security approaches continually evolve [1]. The castle and moat model is a prevalent security solution that establishes network boundaries. It treats the internal Local Area Networks (LAN) as a secure area and ensures safety by blocking external network access. However, the castle and moat model faces challenges in dealing with attacks from devices infected with malware within the LAN [2].

On the other hand, the cloud relay security model represents a different security paradigm that eliminates the need for internal network boundaries. This approach enables secure communication through the cloud without requiring security solutions at the network boundaries. Nevertheless, the cloud relay-based security model carries the risk of information leakage, as the cloud directly handles the client's data [3].

On the other hand, Wide Area Networks (WAN), including remote work utilizing Virtual Private Networks (VPN), have been used to establish communication within LAN [4]. As networks are used in various forms, it becomes crucial to ensure the legitimacy of communication from all directions, regardless of whether the network is internal or external. To address this requirement, a security model called the Zero Trust Model has been proposed as a new security solution [5]. The Zero Trust Model authenticates the legitimacy of all devices, enabling secure communication even in cases of attacks from within the internal network [6]. Additionally, it ensures the security of communication performed by distributed devices on the Internet, such as in remote work or Internet of Things (IoT) systems [7]. These characteristics set it apart from the castle and moat or cloud relay-based model.

The Zero Trust security model necessitates secure connectivity for all devices, even when they are situated in different networks. Consequently, adopting the Zero Trust security model entails addressing two critical issues: connectivity and mobility.

Connectivity refers to establishing communication without impediments from Network Address and Port Translation (NAPT) or differences in IP versions [8], [9]. NAPT is a technology that enables bidirectional translation of IP addresses and port numbers [10]. IP addresses are categorized into global IP addresses and private IP addresses. NAPT conceals devices within the internal network from the external network, making it challenging for devices to initiate communication from outside the NAPT.

IPv6 resolves the limitations of IPv4 concerning the number of assignable IP addresses, which was confined to 2^{32}, by enabling the allocation of 2^{128} addresses [11]. This allows unique address assignment to all devices. However, IPv4 and IPv6 have distinct packet structures, rendering them incompatible and creating challenges in communication. Additionally, since devices still use IPv4 addresses, mixed IPv4 and IPv6 environments are commonplace [12].

Mobility refers to the seamless provision of communication even when the IP address changes. In today's mobile devices, multiple wireless interfaces allow devices to switch interfaces depending on the accessed network [13] [14]. However, the IP protocol does not inherently handle address changes during communication, leading to interruptions in communication when networks switch due to mobility.

The authors have been researching and developing CYber PHysical Overlay Network over Internet Communication (CY-

Authorized licensed use limited to: University of the Philippines - Los Banos. Downloaded on April 29,2024 at 20:19:35 UTC from IEEE Xplore. Restrictions apply.

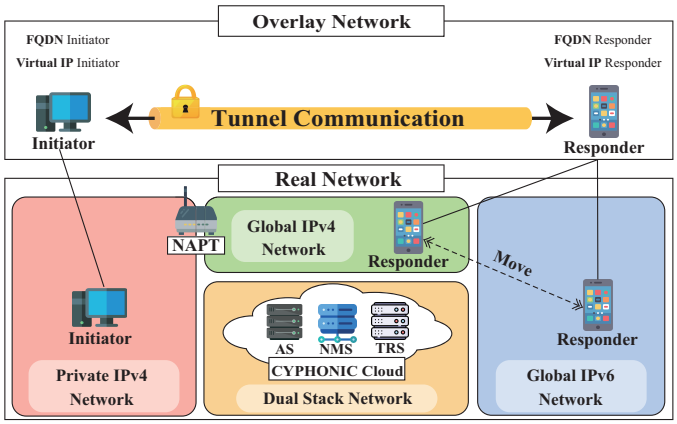

Fig. 1. Overview of CYPHONIC

PHONIC). This secure communication technology achieves both connectivity and mobility based on the concept of zero trust. CYPHONIC establishes an overlay network using virtual IP addresses, enabling encrypted communication exclusively among authenticated devices, ensuring secure communication regardless of the underlying network environment [15], [16]. CYPHONIC consists of CYPHONIC Cloud and CYPHONIC Nodes. Currently, the implementation of CYPHONIC Daemon is limited to devices based on the Linux. However, to make CYPHONIC a more versatile communication technology, it is crucial to extend its support to various platforms.

In this research, the authors aim to achieve a broader usage of CYPHONIC by implementing CYPHONIC Daemon for a new platform, specifically targeting the widely used Windows. Additionally, an installer is introduced to facilitate users in utilizing CYPHONIC. This paper provides detailed information on the installer's design, implementation, and introduction for CYPHONIC Daemon on Windows. Furthermore, the practicality of the proposed system is validated by comparing its performance with the existing CYPHONIC Node.

II. CYPHONIC

Fig. 1 provides an overview of CYPHONIC, which comprises CYPHONIC Cloud and CYPHONIC Nodes. CYPHONIC Cloud offers Authentication Service (AS) responsible for verifying the legitimacy of nodes, Node Management Service (NMS) that handles network information associated with CYPHONIC Nodes and provides opponent information to devices seeking to establish tunnel communication with their counterparts (responders). Additionally, CYPHONIC Cloud includes Tunnel Relay Service (TRS) that facilitates data relay between devices in NAPT routers or across networks with different IP versions.

CYPHONIC Nodes designate responders using the Fully Qualified Domain Names (FQDNs) within their proprietary domain, the CYPHONIC Domain. Since CYPHONIC Nodes use the assigned virtual IP addresses from AS, they achieve communication within the overlay network without being affected by the real network.

During communication, CYPHONIC Nodes establish secure End-to-End (E2E) communication by directly exchanging encryption keys between devices. Moreover, if a CYPHONIC Node's real IP address changes due to mobility, it updates the network information on NMS. Throughout this process, the virtual IP address assigned by AS remains unaffected and unique, enabling seamless communication by specifying the virtual IP address.

The device functionality is realized through an installed terminal program called CYPHONIC Daemon. The CYPHONIC Daemon handles IP datagrams using virtual IP addresses, which applications on CYPHONIC Nodes employ for sending and receiving data. It performs encryption/decryption and encapsulation/decapsulation of datagrams through the virtual interface, thus ensuring secure End-to-End (E2E) communication.

III. Proposed System

A. Overview of proposed system

The CYPHONIC Node incorporates a terminal function known as the CYPHONIC Daemon to facilitate communication within the overlay network. The CYPHONIC Daemon comprises multiple modules that handle different roles, such as interface operations, data encryption, and communication signaling. This design segregates operations that involve manipulating the kernel space from those that only require processes within the user space.

When developing the CYPHONIC Daemon for other operating systems, the focus would primarily be on the modules that interact with the kernel space. In this section, we propose and implement a CYPHONIC Daemon for Windows, specifically targeting the modules that affect the kernel space. Additionally, we introduce an installer to streamline the deployment of CYPHONIC for users.

B. Introducing function

The existing CYPHONIC Node is primarily designed for Linux, and due to the architectural differences between Linux and Windows operating systems, a distinct approach is necessary to handle the kernel space. The following provides a detailed description of the introduced elements.

- Dynamic Link Library (DLL)
  In Windows, a significant portion of the operating system's functionalities is provided through Dynamic Link Libraries (DLLs). Utilizing DLLs offers advantages such as modular coding, code reuse, improved memory usage, and reduced disk space consumption. Furthermore, kernel space operations are performed through DLLs in Windows architecture. Since standard DLLs do not support virtual interfaces, a new DLL must be added to handle virtual interfaces.

Authorized licensed use limited to: University of the Philippines - Los Banos. Downloaded on April 29,2024 at 20:19:35 UTC from IEEE Xplore. Restrictions apply.

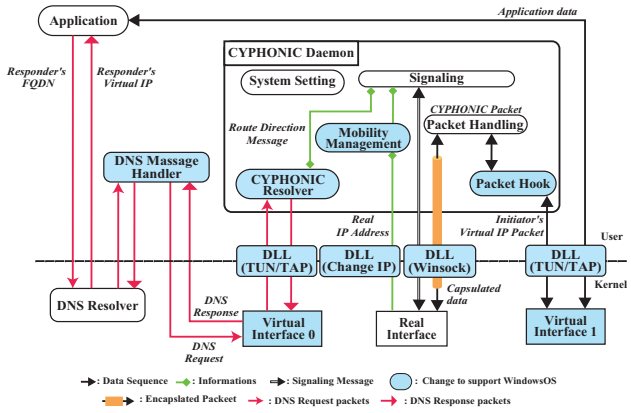

Fig. 2. Overview of CYPHONIC end-device functions on Windows

- Network Socket
  The current implementation for Linux relies on BSD sockets and assumes the usage of virtual interfaces specific to Linux. In this paper, we will design and implement terminal functionality that considers the availability of virtual interfaces for Windows, utilizing Windows network sockets.
- Domain Name System (DNS) Message Handler
  The FQDN used for communication with CYPHONIC includes the CYPHONIC domain. The resolution of FQDNs containing the CYPHONIC domain is carried out through communication with the CYPHONIC cloud. Hence, it is necessary to implement functionality that forwards DNS packets to the CYPHONIC Daemon to resolve DNS queries by collaborating with the CYPHONIC cloud.

C. Overview of CYPHONIC end-device functions on Windows

Figure 2 depicts the proposed CYPHONIC Node system model for Windows. The CYPHONIC Node enables communication within the overlay network by implementing CYPHONIC Daemon and creating virtual interfaces. CYPHONIC Daemon consists of the following modules:

- Configuration Module
  It reads the configuration file required for CYPHONIC communication. The configuration file contains information such as authentication methods, virtual interface settings, and port numbers for communication with CYPHONIC Cloud.
- Signaling Module
  It performs signaling processes, including authentication with the CYPHONIC cloud through AS, the establishment of routes with NMS, and relay processes with TRS.
- CYPHONIC Resolver
  Resolving the FQDN of the CYPHONIC domain through an original DNS server is difficult. Since the resolution from FQDN to a virtual IP address is conducted by signaling with NMS, the CYPHONIC Resolver detects DNS packets and passes them to the signaling module. After completing the signaling process and obtaining the virtual IP address of the responder, the CYPHONIC Resolver generates a DNS response and notifies the virtual IP address to the application.
- Packet Hook Module
  The packets sent to the virtual interface are obtained by the packet hook module. When receiving decapsulated packets from the Packet Handling module, they are forwarded to the application through the virtual interfaces.
- Packet Handling Module
  CYPHONIC performs communication using the defined packet structure. The Packet Handling module is responsible for packet generation, encryption, and decryption. After generating a CYPHONIC packet, a Hash-based Message Authentication Code (HMAC) is attached. This allows the responder to calculate the HMAC of the received packet to ensure integrity.
- Mobility Management Module
  Devices may change their real IP addresses when moving to different network areas. The Mobility Management module detects changes in the real IP address and registers the updated network information with the NMS. This allows the establishment of a tunnel with the responder again, even after the real IP address has changed, enabling seamless communication.

D. Packet flow

1) DNS packet flow: To enable communication within the overlay network, it is essential to respond to DNS queries generated by the application with a response that includes the virtual IP address. NMS maintains a mapping between CYPHONIC's unique FQDN and the corresponding virtual IP address.

Therefore, NMS conducts the resolution of FQDN in CYPHONIC communication. The DNS Message Handler module filters only DNS queries for the FQDN used in CYPHONIC and passes them to the CYPHONIC Resolver module. The CYPHONIC Resolver module initiates the route selection process upon receiving a DNS query and generates a DNS Request to be forwarded to the Signaling module.

The Signaling module sends a Direction Request message to NMS to obtain the network information of the responder. Subsequently, NMS sends a Route Direction to Initiator message to retrieve the virtual IP address of the responder, then passes it to the CYPHONIC Resolver module. Finally, the CYPHONIC Resolver module generates a DNS Response containing the virtual IP address of the responder and returns it to the application.

2) Data packet flow: Packets destined for virtual IP addresses are routed to the virtual interface. When the application sends application data to the responder, the CYPHONIC Daemon captures it through the Packet Hook module and passes it to the Packet Handling module. Then, the virtual IP packet is encrypted, and the CYPHONIC header, which

Authorized licensed use limited to: University of the Philippines - Los Banos. Downloaded on April 29,2024 at 20:19:35 UTC from IEEE Xplore. Restrictions apply.

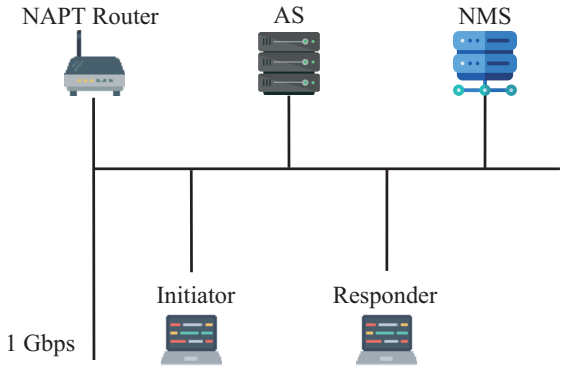

Fig. 3. Evaluation Model

enables CYPHONIC communication, is added to the captured virtual IP packet.

Subsequently, a UDP header is added, and the packet is encapsulated with the real IP address. This process generates a real IP packet, enabling communication through the real interface, which communicates with the responder following the established UDP tunnel.

Upon receiving the packet, the responder removes the real IP header and UDP header, retrieves the CYPHONIC packet, and passes it to the CYPHONIC Daemon. Then, the CYPHONIC packet is decrypted and decapsulated by the Packet Handling module, transforming it into a virtual IP packet. The virtual IP packet is then routed to the virtual interface. Finally, the virtual IP packet is forwarded to the application.

E. Installer

The CYPHONIC Daemon must be installed to enable communication using CYPHONIC. To facilitate this process, we have developed an installer that allows users to install the CYPHONIC Daemon easily. The installer consists of three main components: CYPHONIC EXE, Setup Package, and CYPHONIC Starter Application.

The CYPHONIC EXE contains the binary of the CYPHONIC Daemon. The Setup Package includes modules configured to support the CYPHONIC EXE, providing necessary DLL files for TUN/TAP device operation and containing settings for filter functions that identify domain names.

The CYPHONIC Starter Application prompts users to input their device ID and password. Based on this information, the CYPHONIC EXE is executed. It's important to note that using CYPHONIC requires kernel operations, such as creating virtual interfaces, and can only be achieved with administrator privileges.

IV. Evaluation

We have devised a network configuration to perform operation verification and performance evaluation, as depicted in Fig. 3. Furthermore, Table I provides information about the devices utilized for the performance evaluation. The evaluation criteria for operation verification and performance evaluation are based on the communication performance achieved using CYPHONIC with the existing device functionality designed for Linux.

TABLE I
Details of the Measuring Devices

CYPHONIC Cloud	
OS	Ubuntu 21.10
CPU	11th Gen Intel Core i9-11900K @ 3.50GHz 16cores
Memory	128GB
CYPHONIC Node(Windows)	
OS	Windows 10
CPU	11th Gen Initel Core i5-1135G7@2.40Ghz
Memory	8GB RAM
CYPHONIC Node(Linux)	
OS	Ubuntu 22.04.1 LTS
CPU	11th Gen Initel Core i5-1135G7@2.40Ghz
Memory	8GB RAM

TABLE II
Result of ICMP evaluation

		Windows	Linux
RTT	min	2.0 ms	2.6 ms
	max	8.0 ms	3.8 ms
	avg	3.1 ms	3.1 ms

A. Functional Evaluation

The performance validation was carried out in two scenarios: communication using CYPHONIC with two Windows devices and communication using CYPHONIC with two Linux devices. Table II presents the Round Trip Time (RTT) obtained through ping commands for each case.

The verification results confirmed that communication within the overlay network is achievable, as observed from the successful acquisition of virtual IP addresses based on the Responder's FQDN and the subsequent establishment of connections using these IP addresses. Furthermore, the RTT results confirmed no significant overhead in communication when utilizing Windows devices.

B. Performance Evaluation

We measured throughput TCP and UDP communications using iperf3 in the performance evaluation. The measurements were performed in 10 sets, each measurement was 10 seconds, and the average value of the acquired measurements was computed.

Table III presents the measurement results of TCP performance. The observations indicate comparable communication performance for Windows and Linux devices when utilizing CYPHONIC, confirming sufficient TCP communication performance.

Table IV displays the measurement results of UDP performance. Similar communication performance was observed at 50 Mbps, 60 Mbps, 70 Mbps, and 80 Mbps for both scenarios involving CYPHONIC with Windows devices and CYPHONIC with Linux devices. Additionally, the measured

Authorized licensed use limited to: University of the Philippines - Los Banos. Downloaded on April 29,2024 at 20:19:35 UTC from IEEE Xplore. Restrictions apply.

TABLE III
RESULTS OF TCP EVALUATION

	Windows	Linux
Transfer	0.99 GBytes	2.80 GBytes
Throughput	82.7 Mbps	240.0 Mbps

TABLE IV
RESULT OF UDP EVALUATION

	Windows		Linux	
Traffic	Throughput	Jitter	Throughput	Jitter
50Mbps	49.9 Mbps	0.246 ms	50.0 Mbps	0.198 ms
60Mbps	59.9 Mbps	0.286 ms	60.0 Mbps	0.304 ms
70Mbps	69.9 Mbps	0.276 ms	70.0 Mbps	0.283 ms
80Mbps	79.9 Mbps	0.264 ms	80.0 Mbps	0.153 ms

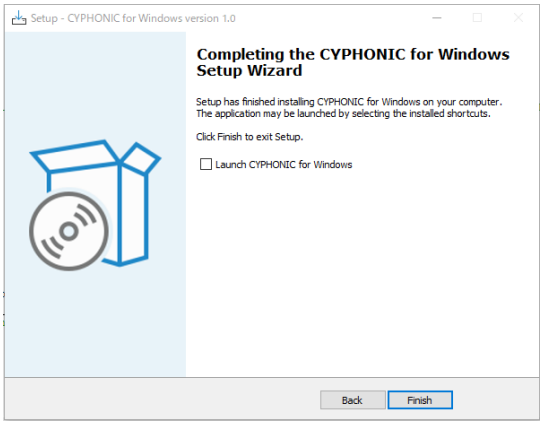

Fig. 4. Result after using installer

jitter values did not significantly impact communication, affirming satisfactory communication performance for UDP.

C. Functional evaluation of installer

The proposed installer was executed on Windows device. The screen image after executing the installer is depicted in Fig. 4. Subsequently, verification was conducted, confirming the successful installation of CYPHONIC EXE, Setup Package, and CYPHNIC Starter Application in the local directory of Windows device. Furthermore, through the execution of the CYPHNIC Starter Application and appropriate input, the successful initiation of the CYPHONIC Daemon was confirmed. Based on these observations, we have reached the conclusion that communication using CYPHONIC is achievable on a Windows device solely through the operation of the GUI application.

V. CONCLUSIONS

This paper proposes a Windows device functionality to enable a more versatile utilization of CYPHONIC. The implementation involves the addition of DLLs that interface with Windows kernel space, allowing for the creation of a virtual interface. This terminal functionality for Windows is achieved without requiring any modifications to the module operating in the user space. Furthermore, an installer tool is provided to facilitate easy adoption of CYPHONIC by users.

The verification and evaluation results demonstrate that the performance of the device functionality for Windows is comparable to that of its Linux counterpart, confirming its practicality. Additionally, the successful operation of the installer tool has been validated, ensuring the broader applicability of CYPHONIC beyond the Linux environment.

ACKNOWLEDGMENT

This work is supported in part by Grant-in-Aid for Scientific Research (C)(21K11877), the Japan Society for the Promotion of Science (JSPS), and the Cooperative Research Project of the Research Institute of Electrical Communication, Tohoku University.

REFERENCES

[1] Y. Zou, J. Zhu, X. Wang and L. Hanzo, "A Survey on Wireless Security: Technical Challenges, Recent Advances, and Future Trends," 10.1109/jproc.2016.2558521, September 2016.
[2] A. Deshpande, "A Study on Rapid Adoption of Zero Trust Network Architectures by Global Organizations Due to COVID-19 Pandemic," 10.9734/bpi/nvst/v1/3640F, August 2021.
[3] B. R. Kandukuri, R. P. V., and A. Rakshit, "Cloud Security Issues," 10.1109/SCC.2009.84, September 2009.
[4] Y. Rekhter and E. C. Rosen, "BGP/MPLS IP Virtual Private Networks(VPNs)," 10.17487/RFC4364, February 2006.
[5] M. Alawneh and I. M. Abbadi, "Integrating Trusted Computing Mechanisms with Trust Models to Achieve Zero Trust Principles," 10.1109/IOTSMS58070.2022.10062269, December 2022.
[6] M. Campbell, "Beyond Zero Trust: Trust Is a Vulnerability," 10.1109/MC.2020.3011081, October 2020.
[7] C. Miller, P. Rodeghero, M. -A. Storey, D. Ford and T. Zimmermann, "Survey Instruments for "How Was Your Weekend?" Software Development Teams Working from Home During COVID-19," 10.1109/ICSE-Companion52605.2021.00101, May 2021.
[8] A. Müller, G. Carle and A. Klenk, "Behavior and classification of NAT devices and implications for NAT traversal," 10.1109/MNET.2008.4626227, September 2008.
[9] C. E. Perkins, "IP Mobility Support," 10.17487/RFC2002, October 1996.
[10] T. Eckert and D. Wing "IP Multicast Requirements for a Network Address Translator(NAT)and a Network Address Port Translator(NAPT)," 10.17487/RFC5135, February 2008.
[11] S. Kawamura and M. Kawashima "A Recommendation for IPv6 Address Text Representation," 10.17487/RFC5952, August 2010.
[12] D. G. Chandra, M. Kathing and D. P. Kumar, "A Comparative Study on IPv4 and IPv6," 10.1109/CSNT.2013.67 April 2013.
[13] P. Ferrari, A. Flammini, D. Marioli and A. Taroni, "IEEE802. 11 sensor networking," 10.1109/TIM.2006.870105, March 2006.
[14] D. Saliba, R. Imad, S. Houcke and B. E. Hassan, "WiFi Dimensioning to offload LTE in 5G Networks," 10.1109/CCWC.2019.8666585, January 2019.
[15] T. Yoshikawa, H. Komura, C. Nishiwaki, R. Goto, K. Matama and K. Naito, "Evaluation of new CYPHONIC: Overlay network protocol based on Go language," 10.1109/ICCE53296.2022.9730323, January 2022.
[16] T. Yoshikawa, S. Isomura, H. Komura, S. Kubota, C. Nishiwaki and K. Naito, "Performance evaluation of shared library supporting multi-platform for overlay network protocol," 10.1109/GCCE50665.2020.9292015, October 2020.

Authorized licensed use limited to: University of the Philippines - Los Banos. Downloaded on April 29,2024 at 20:19:35 UTC from IEEE Xplore. Restrictions apply.