Remote Power Control and Monitoring of Cross-Platform Devices Using Magic Packets Across Network Boundaries

Christian von Arnim*, Evelyn Dalkowski*, Tobias Bodamer*, Ziya Selim*,
Carsten Ellwein* and Andreas Wortmann*
*Institute for Control Engineering of Machine Tools and Manufacturing Units
University of Stuttgart, Germany
Email: {christian.von-arnim, carsten.ellwein, andreas.wortmann}@isw.uni-stuttgart.de

2024 IEEE 3rd Industrial Electronics Society Annual On-Line Conference (ONCON) | 979-8-3315-4031-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/ONCON62778.2024.10931695

Abstract—**This paper compares online device discovery and remote startup and shutdown solutions for an enterprise network. As some of for these solutions do only work in a local network and the enterprise network consists of several subnetworks with different firewalls setting, a concept to communicate securely across subnetworks is required. For an evaluation, a prototype is implemented using Wake-on-Lan for startup, Address Resolution Protocol for online detection and Secure Shell for shutting down. Bridged MQTT-brokers are used to communicate across subnetwork with a service in every subnetwork. The central hosted prototype is able to switch on or off computers remotely in every subnetwork.**

I. Introduction

Power consumption in corporate and enterprise networks continues to grow as the number of machines deployed in these environments increases. As a result, minimizing energy usage has become a primary goal for many organizations. Various studies, such as [1]–[3] have explored different methods to decrease the electricity usage of devices in a network. A reduction of electricity can be achieved by shutting down devices when they are not in use, thereby creating manual non-operational periods to save energy. This can be achieved by implementing the ability to remotely shutdown or startup devices and machines that are currently not in use. The objective of this paper is to develop a system that facilitates remote shutdown and startup of devices within a network. To achieve this, it is crucial to first detect the presence of devices within the network to get an overview of the currently running machines. Therefore the system also includes an online detection mechanism to identify whether a device is currently powered on or off and to enable appropriate shutdown or startup actions based on the device's status. In addition, the online detection mechanism introduces a method for scanning the network for foreign devices, thereby enhancing network security by identifying unauthorized devices promptly. As our corporate network includes machines running either Windows or Linux operating systems, we need to develop a system that is compatible with both.

Due to security concerns, corporate and enterprise facilities separate their networks into multiple subnetworks. Sending instructions to a target device in a different subnetwork presents a significant challenge. This is because these subnets enforce security policies at their boundaries to encapsulate applications used within the network by setting up a firewall at every border. Communication between the networks is therefore restricted. In our case, a centrally hosted hardware management system should be able to command all devices on other networks. Nonetheless, we need to communicate across a network boundary that has been limited by a firewall. To address this issue, it is imperative to develop a method for communicating across network boundaries without introducing new security vulnerabilities. Therefore, this work also explores technologies that enable secure communication across these boundaries.

In the following Section II, available technologies that can be used to enable the shutdown, startup, and online detection of devices will be explored. Section III lists related work. In Section IV, the proposed architectural design, along with the system requirements within the current network, will be delved into. Then, in Section V, the technologies introduced in Section II that fulfill these requirements will be identified and evaluated. Finally, a proof of concept will be presented in Section VI.

II. Foundations

This chapter presents current technologies for shutdown and startup, online detection, and communication across network boundaries, and explain features required for our system which are discussed and compared in more detail in the tables in Section V.

A. Startup and Shutdown

Wake-on-LAN (WoL) is an established industry protocol [4] for waking up computers over a network. It is implemented using the Magic Packet Technology [5] which uses an Ethernet frame with a specific payload containing of the target device's hardware address. The network interface controller (NIC) of the device scans incoming frames and wakes it up if the address matches. For this to function, the NIC must support WoL, have it enabled, and be in a low-powered state. To use WoL across subnets [6], the magic packet must be sent to the target subnet as a subnet-directed broadcast [5] using a routed, connectionless protocol like the user datagram protocol (UDP) [7]. Broadcast forwarding must be enabled on the router, and the firewall has to be configured to allow incoming Magic Packets[1]. There is no authentication, no authorization, no acknowledgement, and no security. WoL is not reliable, as there are other factors, such as the power state, which determines whether the command is executed or not.

Intel Active Management Technology (AMT)[2] is part of the Intel vPro platform and provides remote out-of-band management for devices, including remote power control with Transport Layer Security (TLS) [8] encryption using the Transmission Control Protocol (TCP) [9], [10]. Compared to other startup technologies, AMT offers security against unauthorized access through the use of authentication and authorization[2]. Intel AMT also encrypts its TLS connection[2] using public-key infrastructure [11]. This technology gives feedback on whether a computer has started, as it has a management client that monitors the devices[2]. But Intel AMT requires vPro enabled hardware that not every commercially available computer has[2].

Remote procedure calls (RPCs) [12] is a concept that is used for inter-process communication within a network. A procedure with parameters can be sent by a caller to a server via a network. The necessary parameters for executing the procedure and the name of the procedure are passed via the call. After finishing the process, the server sends a reply message, which contains the results of the procedure [13] and has therefore some sort of feedback. No technology for using RPC is pre-installed on Linux, so you have to install one first. Samba, for example, implements RPC for Linux operating systems. RPC uses authentication and authorization to prevent unauthorized access, but it does not check whether the message was altered and thus poses a security risk[3].

Secure Shell (SSH) [14] is a cryptographic network protocol that allows secure remote login and command execution on machines in an insecure network. After authentication and authorization of the client via, e.g., password or public key authentication [14]. It can be used to execute shutdown commands on remote machines to power them off. Client and server applications are available for both Linux and Windows systems. As SSH is executed in a command line interpreter [14], the user also receives feedback as to whether his commands are being executed.

B. Online Detection

Internet Control Message Protocol (ICMP) for IPv4, as defined in RFC 792 [15], is part of the TCP/IP suite and an integral part of Internet Protocol (IP) [16]. One type of ICMP message is an echo request, commonly known as "ping", which can be used to determine whether a network device is reachable. ICMP can be used to detect devices on a different network [15] if it is accessible. An ICMP echo request message is sent to the target host, which replies with an ICMP echo reply, if reachable. The network device has to be configured to reply to or forward the ICMP echo requests. This could lead to potential security risks, which, however, do not pose a major threat to our setting and are therefore negligible. There is also no authentication or authorization. It runs on both Windows and Linux. It should also be noted that ICMP can be switched off. In Windows 11 systems, ICMP is blocked by the Windows firewall[4].

Address resolution protocol (ARP), as defined in RFC 826 [17], is a network protocol that determines a network address for an IPv4 address of the Internet layer and stores this assignment in the ARP tables of the participating computers if necessary. To determine the associated MAC address for an IPv4 address in the local network, an active ARP request is sent as a data link layer [18] multicast. If a device with this IPv4 address is available, it need respond to this request. It is also possible for the network devices to listen passively to ARP requests that are frequently sent around the network to get a list of available devices in the network. To detect incorrect MAC addresses, ARP can be used since the messages contain the sender's MAC and IP addresses. ARP is used on Windows and Linux and as with ICMP, ARP has no authentication or authorization.

C. Communication across network boundaries

Transmission Control Protocol/Internet Protocol (TCP/IP) [19] refers to a group of network protocols, including TCP and IP. These protocols are a standard for communication in networks. TCP, an event-based communication technology [10], is used to establish a connection between the devices involved [19]. As the data is transmitted via TCP, bidirectional communication is supported for this transmission [10]. The message is then sent to the recipient via the IP address [19]. TCP/IP works independently of the operating system, which means that the underlying system can be easily expanded.

Hypertext Transfer Protocol (HTTP) [20] is a protocol that is used to transfer data via a computer network. A connection is established between client and server, which the client initiates to send a message. The latest version of HTTP is HTTP/3 [21], which allows bidirectional communication. Asynchronous communication is also possible via server sent events. Hypertext Transfer Protocol Secure (HTTPS) [22] was invented for secure communication on the World Wide Web. Here, data is transmitted in form over an encrypted TLS channel. HTTPS also has an authentication and an authorization method.

Message brokers [23] help with the communication between sender and receiver by converting the message into the respective message protocol before passing it on. There are various message brokers, e.g., MQTT, which we will explain in more

[1]https://www.cisco.com/c/en/us/support/docs/switches/catalyst-3750-series-switches/91672-catl3-wol-vlans.html

[2]https://software.intel.com/sites/manageability/AMT_Implementation_and_Reference_Guide/default.htm

[3]https://learn.microsoft.com/en-us/windows/win32/rpc

[4]https://learn.microsoft.com/en-us/windows/security/operating-system-security/network-security/windows-firewall/configure

detail here. MQTT [24] is a bidirectional, event based transport protocol that is used to publish and subscribe to messages by using TCP/IP to transfer data. In using MQTT, a client opens a connection to the broker to send messages that the client wants to spread. The client can also subscribe to request application messages from the broker that it wants to receive. The broker accepts the network connections and the published messages of the clients and redistributes them. MQTT offers an implementation for authorization and authentication and can be encrypted [24]. An MQTT bridge [25] connects two MQTT brokers that they can exchange messages. The MQTT broker also acts as a client through the MQTT bridge and publishes data from other brokers to subscribers or sends its own messages to other brokers.

SignalR [26] is an open-source library for Microsoft ASP.NET that provides a simple interface for using RPCs. It allows server code to be sent asynchronously to web applications by using HTTP(S) polling or WebSockets for bidirectional communication. Sending code asynchronously is useful because the server does not have to wait for a client request in order to send a new message, resulting in a faster exchange. The security functions of SignalR are authentication, authorization, and connection tokens.

A virtual private network (VPN) [27] is used to set up a private tunnel on a public network in order to encrypt communication. A network tunnel is built into the public network for this purpose. The VPN tunnel allows an authenticated device to connect virtually to a network and thus obtain the same rights as a device that is connected to the network.

SSH tunneling can be used to forward TCP/IP traffic on specific ports from the remote host to the client behind firewall, or Network address translation (NAT) [14], which is called a reverse tunnel. Importantly, this procedure necessitates no further alterations to firewall configurations. However, it is imperative to acknowledge the inherent risks associated with this approach. If access to the local machine is compromised, it potentially grants access to all machines within the otherwise inaccessible network as well. Unlike other technologies presented, this is neither bidirectional nor event-based.

III. Related Work

In [28], multiple problems that can arise when trying to implement a way to remotely startup desktop devices via WoL technology in institutional networks are highlighted. To solve these problems, a solution that doesn't require the use of a local agent installed on all client computers is explored in this paper. The creation of a firewall port that allows the sending of WoL packages to start up or shut down devices through the network boundaries was involved in their solution. As mentioned in section I their solution is not feasible in our case since this approach could introduce a safety concern in the network structure.

In [29], end-to-end communication using the MQTT technology as a replacement for the Client/Server paradigm is the focus of the research. A network architecture, in which both the client and the server use the MQTT protocol to publish messages to the MQTT Broker positioned between them, is created. The message is then forwarded by the MQTT Broker to the corresponding client. As this is only an approach for communication, the evaluation of this technology based on our requirements for the system will be necessary. Depending on the results, this approach can be a potential solution for our communication issue.

Currently, a single solution that can remotely shut down, start up, and detect devices that are online doesn't exist, and therefore the development of our own solution is required.

IV. Architectural design

This section describes the architecture of the system shown in Fig. 1. A hardware management system is deployed in

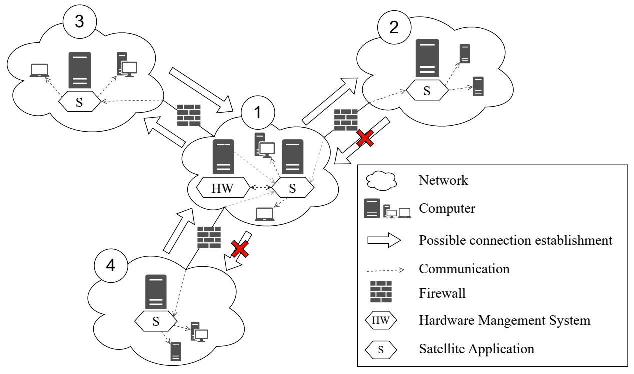

Fig. 1. Architectural design of the system for communication across network boundaries with satellites

the corporate network (1), and should be able to manage machines in multiple networks. Certain networks, such as sub-organizational networks (3), facilitate largely unrestricted bidirectional communication (establishment) between the corporate network (1), while others have restricted communication in one direction. Devices within a network which acts as a demilitarized zone (DMZ) (2) are accessible from external networks, yet unable to communicate with hosts outside this network. Conversely, devices within a network situated behind a router employing one-to-many network address translation (NAT), such as those found in operational technology (OT) networks (4), can access hosts in external networks but are not individually accessible from external networks. The goal of this paper is to find a solution that enables the objectives not only for the current network infrastructure but also for any future network, wherein the devices can either directly reach or be reached by the hardware management system in network (1). Within each managed network, we can deploy a so called satellite — an application designed to facilitate all necessary functionalities, including startup, shutdown, and online detection procedures for machines within that network. By establishing connections between these satellites and the system, we can create secure and tight communication channels between networks, tailored to our requirements. This architecture allows commands such as startup, shutdown, and online detection from the system to be relayed by the satellites to the intended machines and vice versa.

In the upcoming chapter, a secure communication protocol will be selected to be utilized between satellites to abstract the functionalities. Additionally, a specific technology will be chosen to implement each of the functionalities mentioned above on satellites.

V. Comparison of technology

The technologies in Section II are evaluated based on the following requirements for the system. Each technology will be assigned a rating according to its level of fulfillment of these requirements. The requirements and the corresponding technologies are divided into two groups. The first group, *general criteria*, applies to all technologies relevant to startup, shutdown, and online detection. The second group, *communication criteria*, applies to all technologies that can be used for the proposed satellite communication. The evaluation of the general criteria are shown in Table I and that of the communication criteria in Table II.

The first criterion, cross platform availability, evaluates the extent to which the technology can be used on different operating systems and hardware technologies. As mentioned in Section I, the primary requirement for the solution is usability across multiple operating systems. As it is imperative that no new security vulnerabilities are introduced in the network, the technologies are evaluated if they have security features. If the usage of these technologies does not introduce a new vulnerability they will fulfill this criterion fully. Acknowledgement refers to the sending of a receipt message to the sender to indicate that the requested action has been executed. The final criterion is reliability, which assesses whether obstacles such as firewalls could impede the connection. These two criteria ensure the arrival of instructions at the target device.

For the same security reasons, encryption is the first criterion for the communication technologies. Bidirectional communication between the satellites and the clients is crucial for exchanging messages in the aforementioned architecture. The next criterion is event-based communication, which is important for keeping the message channel open to save time when exchanging messages. Expandability assesses the system's ability to accommodate new communication partners in the future, authentication and authorization are criteria important for both groups as they prevent unwanted use.

In the following, we will take a closer look at the technologies and use the criteria to explain why we decided for the technologies for our system.

There are two technologies that we have taken into account for booting up the computers remotely: Wake-on-LAN and Intel AMT. Considering our criteria, Wake-on-LAN satisfies cross-platform availability since it operates on any standard computer. WoL does not provide features for the other criteria, thus they remain unfulfilled. Reliability is not met because factors such as power status determine whether the command is executed. Therefore, Intel AMT offers many advantages over WoL, fulfilling every criterion except cross platform availability, as it does only work on Intel Plattforms with Pro enabled hardware. As this is the main requirement, WoL was selected for the implementation. In our evaluation of technologies for remote shutdown, we compared RPC and SSH in table I. Both technologies have cross platform availability and use authentication and authorization methods. When examining the security and acknowledgment criteria, SSH meets both, whereas RPC has a security concept that is vulnerable to attacks. Therefore, based on our evaluation, SSH emerges as the better choice, providing a more secure solution.

For online detection, we have examined two technologies in more detail: ICMP and ARP. Both technologies do not have authentication or authorization features. And both provide feedback on a ping attempt as to whether they have reached a host. However, ARP is more reliable than ICMP for data transmission because ICMP can be disabled. Therefore, ICMP was disregarded in favor of ARP.

TCP/IP fulfills all our other criteria but lacks authentication, authorization, and encryption features, thus it is ruled out. REST fulfills almost all criteria, at least partially, but is excluded due to its lack of support for event-based communication. SignalR fulfills all our criteria completely, except for expandability, which is only partial, and event based communication, which is not at all. For the technology VPN, we were unable to apply every criteria except of authentication, which is why we decided against this technology as well. SSH tunneling also meets every criteria except bidirectional and event-based communication is not implemented in the standard version which is why we do not use this technology either. The Broker Technology is used in the system, as it is the only technology that fulfills all our criteria. MQTT has been selected as the representative for broker technology due to its bridging capabilities.

VI. Proof of Concept

To implement and verify our solution in a test network as described in section IV, we developed a system that integrates SSH for remote shutdown, WoL for startup, and ARP for online detection. Four Linux machines, running Debian 12, were deployed one in each network as outlined in section IV and equipped with MQTT brokers (Version 2.0.11). SSL and client certificate authentication were enabled on each broker using different certificates for each satellite. A unique network ID was assigned to each satellite as an MQTT prefix, and topics were crafted for each functionality with the structure `networks/<id>/<service-name>`. MQTT bridges were used between the main broker (in the corporate network) and the brokers in the other networks to forward messages across networks. Bridge connections were established in the direction where connection establishment was not restricted, and topic filters were configured on each bridge to allow only required messages with the specific network ID to be forwarded in both directions.

Additionally, a .NET 8.0 service implementing the core functionalities, was deployed to each satellite. This service uses the MQTT.NET library (4.3.6.1152) to connect

TABLE I
COMPARISON OF DIFFERENT TECHNOLOGIES FOR START-UP, SHUT-DOWN AND ONLINE DETECTION

	Start-Up		Shut-Down		Online-Detection	
Criteria \ Technologies	WoL	Intel AMT	RPC	SSH	ICMP	ARP
Cross Platform Availability	●	○	●	●	●	●
Security	○	●	◑	●	◍	◍
Authentication	○	●	●	●	○	○
Authorization	○	●	●	●	○	○
Acknowledgement	○	●	◑	●	●	●
Reliability	○	◍	◍	◍	○	●

● Completely Fulfilled ◑ Partially Fulfilled ○ Not Fulfilled ◍ Not Applicable

TABLE II
COMPARISON OF DIFFERENT TECHNOLOGIES FOR COMMUNICATION

Criteria \ Technologies	TCP/IP	REST	Broker Technology	SignalR	VPN	SSH Tunneling
Encrypted	○	●	●	●	◍	●
Bidirectional Communication	●	◑	●	●	◍	○
Event Based Communication	●	○	●	○	◍	○
Expandability	●	●	●	◑	◍	○
Authentication	○	●	●	●	●	●
Authorization	○	●	●	●	◍	●

● Completely Fulfilled ◑ Partially Fulfilled ○ Not Fulfilled ◍ Not Applicable

to the local MQTT broker using the same certificate and subscribes to all topics with the assigned network ID (`networks/<id>/#`). The implementation of online detection utilizes SharpPcap 6.3.0, which relies on libpcap on Linux, to passively listen to and actively send ARP requests. For remote startup the library WakeOnLAN, version 2.0.2 was used. The shutdown implementation establishes an SSH connection (SSH.NET library, 2024.0.0) with the target host using a shared private key and executes the shutdown command. The exit code is evaluated to acknowledge the success of the shutdown and published to the response topic.

To enable the shutdown functionality, a common RSA key was deployed to all machines, the OpenSSH server was activated and permissions for the execution of the shutdown command where granted.

To test the functionality of this system, a .NET 8.0 Tester console application was implemented. It was connected to the main broker, subscribed to all topics (`networks/#`) and calls the services for WoL, Shutdown and online detection.

Using this tester the following results were obtained: In the corporate and sub-organizational networks, the test passively detected ARP requests. WoL was successfully tested with one desktop machine and a laptop, where WoL was enabled in the BIOS beforehand and the latter connected to the power outlet. Shutdown was successfully tested on Windows 10, with a user set up for SSH shutdown as described above. Shutdown was successfully tested with a Linux machine using the hostname of that machine.

In the OT network, shutdown of a Linux machine was tested using the fully qualified domain name, which was successful. Startup was tested on the same machine without further configuration, which was not successful. Due to no network activity, no passive ARP request was detected over 30 minutes, but actively sending ARP requests resulted in 3 Linux machines immediately responding.

In the DMZ network, ARP requests were captured passively and a Windows machine responded to active ARP requests. Shutdown was successfully tested with the same laptop running Windows 10 mentioned above using the IP address of the device. WoL was tested with this laptop in hibernation first, which was not successful. Repeating the test in the powered off state was successful.

VII. CHALLENGES AND OPPORTUNITIES

While testing the implementation some obstacles came up with the usage of WoL. First of all, WoL needs to be enabled in the BIOS of every device you want to wake up remotely. When a device is first set up, the BIOS has to be configured anyway, so enabling WoL for new devices can be done with little effort. However, checking this configuration on all the existing devices in a network can be quite tedious. A second problem is the remote startup of laptops, as the tested laptops needed to be connected to the AC adapter to process the WoL request, and be in a specific power state. From our test results we conclude that online detection using ARP works without compromises, given the ability to actively send such requests in cases of low network activity. Remote shutdown was effectively enabled on both Windows and Linux machines using SSH with passwordless authentication. WoL was successfully triggered on all tested target machines with appropriate BIOS settings, demonstrating reliable remote startup capabilities when configured correctly. Using multiple MQTT brokers in the satellite architecture with secure communication has proven to be reliable and extensible.

Security of SSH for shutdown could be further improved by restricting the executable commands. One limitation of these tests is the use of a small number of test devices and operating systems, although we aimed to provide as much variety as possible. Furthermore, one can investigate whether MQTT is the best broker technology or whether another technology with a self-built bridge would be more efficient. Additional technologies could also be implemented in a second system to provide a backup in case the first technology fails. IPv6 support for online detection could also be included.

VIII. Summary and Conclusion

This paper examines technologies that enable remote control of devices across network boundaries while maintaining security requirements. Such network boundaries restrict the communication establishment between some networks in one direction and also prevent broadcast and non-IP communication between different networks. While existing research has not provided a comprehensive solution, it has highlighted the use of broker technologies to facilitate end-to-end communication across such boundaries. A so-called satellite architecture was proposed to bridge the network boundaries. This approach involves deploying an application in each network to locally enable remote control features and establishing secure communication channels between these applications, thereby creating a secure mesh across the networks. The identified technologies for satellite communication, startup, shutdown and online detection where compared and evaluated under various criteria. For each use case, the most suitable technology was determined: MQTT for satellite communication, SSH for remote shutdown, WOL for remote startup, and ARP for detecting network devices. A proof of concept was created by developing satellite applications and deploying them in a test network. Final tests demonstrated that cross-network communication worked flawlessly, and remote control features operated successfully across different platforms, with only minor challenges related to device setup. The shown architecture fulfills the goal of saving energy by shutting down computers when they are no longer needed. In the context of this paper, tests have only been carried out with a small number of devices and systems. For a more comprehensive statement on functionality, structured tests with different device classes, chip generations and operating systems are required. A security analysis, where the security is investigated with different attack vectors, is also pending.

Acknowledgements

This work is partly funded by the joint project SDM4FZI, supported by the Bundesministerium für Wirtschaft und Klimaschutz (BMWK, Federal Ministry for Economic Affairs and Climate Action) as part of the "Future Investments in the Automotive Industry" funding program.

References

[1] C. Gunaratne, K. Christensen, and B. Nordman, "Managing energy consumption costs in desktop PCs and LAN switches with proxying, split TCP connections, and scaling of link speed," *International journal of network management*, vol. 15, no. 5, pp. 297–310, 2005.

[2] L. Chiaraviglio, M. Mellia, and F. Neri, "Reducing power consumption in backbone networks," in *2009 IEEE international conference on communications*. IEEE, 2009, pp. 1–6.

[3] K. J. Christensen, C. Gunaratne, B. Nordman, and A. D. George, "The next frontier for communications networks: power management," *Computer Communications*, vol. 27, no. 18, pp. 1758–1770, 2004.

[4] K. Kumar, K. Patange, P. Pete, M. Wankhade, A. Chatterjee, and M. Kurhekar, "Power and energy-efficient VM scheduling in OpenStack cloud through migration and consolidation using Wake-on-LAN," *IETE Journal of Research*, vol. 69, no. 11, pp. 8045–8057, 2023.

[5] "Magic packet technology," https://www.amd.com/content/dam/amd/en/documents/archived-tech-docs/white-papers/20213.pdf, accessed: 2024-04-29.

[6] "Internet Standard Subnetting Procedure," DOI: 10.17487/RFC0950, 1985.

[7] "User Datagram Protocol," DOI: 10.17487/RFC0768, 1980.

[8] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," DOI: 10.17487/RFC8446, 2018.

[9] R. T. Braden, "Requirements for Internet Hosts - Communication Layers," DOI: 10.17487/RFC1122, 1989.

[10] W. Eddy, "Transmission Control Protocol (TCP)," DOI: 10.17487/RFC9293, 2022.

[11] D. W. S. Ford, D. S. Chokhani, S. S. Wu, R. V. Sabett, and C. C. R. Merrill, "Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework," DOI: 10.17487/RFC3647, 2003.

[12] P. G. Soares, "On remote procedure call," in *Proceedings of the 1992 conference of the centre for Advanced Studies on Collaborative research-Volume 2*, 1992, pp. 215–267.

[13] "RPC: Remote Procedure Call Protocol specification: Version 2," DOI: 10.17487/RFC1057, 1988.

[14] C. M. Lonvick and T. Ylonen, "The Secure Shell (SSH) Connection Protocol," DOI: 10.17487/RFC4254, 2006.

[15] "Internet Control Message Protocol," DOI: 10.17487/RFC0792, Sep. 1981. [Online]. Available: https://www.rfc-editor.org/info/rfc792

[16] D. S. E. Deering and B. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," DOI: 10.17487/RFC8200, 2017.

[17] "An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware," DOI: 10.17487/RFC0826, 1982.

[18] J. Conard, "Services and protocols of the data link layer," *Proceedings of the IEEE*, vol. 71, no. 12, pp. 1378–1383, 1983.

[19] C. J. Kale and T. J. Socolofsky, "TCP/IP tutorial," DOI: 10.17487/RFC1180, 1991.

[20] R. T. Fielding and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content," RFC 7231, Jun. 2014. [Online]. Available: https://www.rfc-editor.org/info/rfc7231

[21] M. Bishop, "HTTP/3," DOI: 10.17487/RFC9114, 2022.

[22] R. T. Fielding, M. Nottingham, and J. Reschke, "HTTP semantics," *RFC 9110*, 2022.

[23] K. Apshankar, H. Chang, M. Clark, E. B. Fernandez, P. Fletcher, W. Hankison, J. J. Hanson, R. Irani, K. Mittal, J. M. Myerson *et al.*, *Web Services business strategies and architectures*. Springer, 2002.

[24] "MQTT version 5.0," https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html#_Toc3901018, accessed: 2024-04-18.

[25] F. Buccafurri, V. De Angelis, and S. Lazzaro, "MQTT-A: A broker-bridging P2P architecture to achieve anonymity in MQTT," *IEEE Internet of Things Journal*, 2023.

[26] "Introduction to SignalR," https://learn.microsoft.com/en-us/aspnet/signalr/overview/getting-started/introduction-to-signalr, accessed: 2024-04-18.

[27] Z. Xu and J. Ni, "Research on network security of VPN technology," in *2020 International Conference on Information Science and Education (ICISE-IE)*. IEEE, 2020, pp. 539–542.

[28] P. Luberus and A. Nyandoro, "Implementing Wake-on-LAN in institutional networks," *Journal of Applied Business and Economics*, vol. 16, no. 1, pp. 66–73, 2014.

[29] Y. Tang, F. Wu, Z. Liu, and W. Mai, "Research on NAT Traversal Communication based on MQTT," in *Proceedings of the 2021 9th International Conference on Communications and Broadband Networking*, 2021, pp. 186–191.