Journal of Computer and Communications, 2024, 12, 81-98
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Optimizing Resource Management for IoT Devices in Constrained Environments

Sadia Islam Nilima[1], Md Khokan Bhuyan[2], Md Kamruzzaman[3*], Jahanara Akter[3], Rakibul Hasan[3], Fatema Tuz Johora[3]

[1]Department of Business Administration, International American University, 3440 Wilshire Blvd STE 1000, Los Angeles, CA 90010, United States
[2]Department of Technology & Engineering, Westcliff University, 17877 Von Karman Ave 4th Floor, Irvine, CA 92614, United States
[3]Department of Business Administration, Westcliff University, 17877 Von Karman Ave 4th Floor, Irvine, CA 92614, United States
Email: *m.kamruzzaman.130@westcliff.edu

How to cite this paper: Nilima, S.I., Bhuyan, M.K., Kamruzzaman, M., Akter, J., Hasan, R. and Johora, F.T. (2024) Optimizing Resource Management for IoT Devices in Constrained Environments. *Journal of Computer and Communications*, **12**, 81-98.
https://doi.org/10.4236/jcc.2024.128005

Received: July 14, 2024
Accepted: August 19, 2024
Published: August 22, 2024

Abstract

Embedded computing device implementation on the Internet of Things (IoT) requires careful assessment of their intrinsic resource limitations. These constraints are not limited to memory and processing capabilities but extend to the network interfaces, particularly due to the low-power radio standards that these devices typically employ. The IPv6 protocol is shown to be a strong option for guaranteeing interoperability in the IoT, mostly because of its large address space, the range of current IP-based protocols, and its intrinsic versatility. Considering these benefits, we investigate if current IP-based network management protocols can be implemented on devices with limited resources. We investigate the resource needs in particular for implementing Network Configuration Protocol (NETCONF) and Simple Network Management Protocol (SNMP) on an 8-bit AVR-based device. Our investigation reveals the specific memory and processing demands of these protocols, providing valuable insights into their practicality and efficiency in constrained IoT environments. This study underscores the potential and challenges of leveraging IPv6-based network management protocols to enhance the functionality and interoperability of IoT devices while operating within stringent resource limitations.

Keywords

Internet of Things, Resource Constraint, IPv6 Protocol, IP-Based Network Management, Network Management Protocol, Network Configuration Protocol

1. Introduction

The widespread availability of cost-effective embedded computing devices,

equipped with wireless communication capabilities, marks a significant catalyst for the evolution of the IoT. This transformative trend is poised to revolutionize diverse sectors by facilitating the interconnection of myriad devices and systems. In the realm of home automation, for instance, IoT technologies promise seamless integration of household appliances, lighting, security systems, and environmental controls, enabling enhanced convenience, efficiency, and security for homeowners [1]-[10]. Moreover, in the context of energy management, the IoT holds immense potential for optimizing electricity grids through the deployment of smart meters, sensors, and actuators [11]. These interconnected devices can facilitate real-time monitoring, analysis, and control of energy consumption, thereby fostering a more sustainable and resilient energy infrastructure. Beyond these applications, the IoT ecosystem offers fertile ground for the convergence of existing Internet-based software services with the data-gathering and processing capabilities of embedded devices [12]. This convergence is anticipated to give rise to a rich array of innovative computing services, spanning sectors such as healthcare, transportation, agriculture, and manufacturing. Ultimately, the IoT promises to not only enhance the efficiency and functionality of existing systems but also to catalyze the development of entirely new paradigms, shaping the future landscape of technology and society in profound ways.

In the rapidly evolving landscape of IoT deployment, particularly in highly competitive markets, the trajectory of technological advancement, in line with Moore's Law, is poised to drive significant improvements in embedded devices [13]-[17]. These developments should mostly concentrate on making gadgets smaller, more energy-efficient, and less expensive; increasing processing power may not be as important. Typical IoT devices are often powered by 8- or 16-bit microcontrollers, characterized by modest RAM and storage capacities. IEEE 802.15.4 radios are included into various devices to efficiently overcome these resource limitations. This allows low-power, low-data-rate wireless personal area networks (WPANs) with frame sizes of up to 127 octets and data rates ranging from 20 to 250 kbps [18]. The Internet Engineering Task Force (IETF) created the 6LoWPAN standard to facilitate smooth incorporation inside the larger Internet ecosystem. Through the function of an adaptive layer, this standard makes IPv6 packet exchange over IEEE 802.15.4 links possible, opening the door to effective and interoperable communication in Internet of Things networks [19]-[21]. **Figure 1** exhibits the background of IoT. Managing resource-constrained networks within the Internet of Things (IoT) framework is pivotal for ensuring efficient operations and optimal utilization of resources. In this context, the selection of appropriate network management protocols becomes crucial. Our study focuses on implementing the Simple Network Management Protocol (SNMP) [22] and NETCONF [23] [24] protocols, both recognized for their efficacy in managing network resources. SNMP, a widely adopted protocol, offers a straightforward approach to network management, utilizing a hierarchical structure and standardized MIBs (Management Information Bases) for moni-

toring and configuration tasks. Conversely, NETCONF, built upon a more modern architecture, employs XML-based data encoding and operates over SSH (Secure Shell) or TLS (Transport Layer Security), facilitating secure and efficient communication. By implementing both protocols, we aim to conduct a comparative analysis of their resource utilization characteristics, enabling us to discern the strengths and limitations of each approach. This comparative evaluation will provide valuable insights into selecting the most suitable protocol for managing resource-constrained networks within the IoT ecosystem, thereby contributing to the advancement of network management practices in such environments.

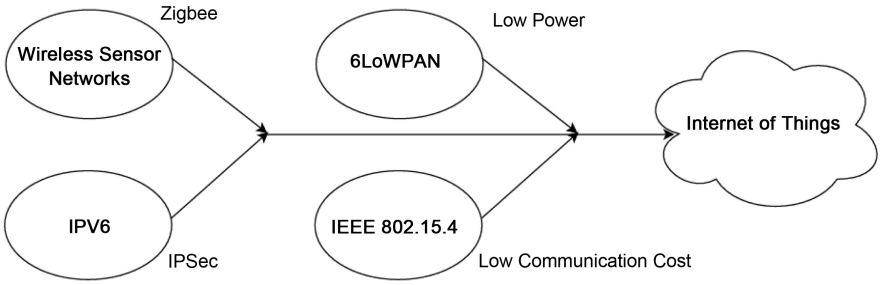

Figure 1. Background of IoT [25].

In this paper, we embark on a comprehensive exploration of resource-constrained hardware, laying the foundation to comprehend the intricate limitations it imposes. Our journey begins with an incisive overview, meticulously dissecting the unique challenges inherent in such environments. Through this lens, we gain invaluable insights into the nuanced landscape where efficiency is paramount and resources are scarce. Subsequently, we pivot towards a meticulous exposition of our chosen development environment, meticulously crafted to accommodate the implementation of two pivotal management protocols: SNMP and NETCONF. Within these protocols lies the cornerstone of network administration, and as such, their integration into resource-constrained devices demands meticulous attention to detail and adept navigation of limitations. Delving deeper, we unveil the intricacies of their security mechanisms, underscoring the imperative of safeguarding network integrity and confidentiality amidst resource constraints. Finally, as our journey draws to a close, we reflect upon the rich tapestry of experiences garnered through the implementation of these protocols, offering a candid discourse on the challenges surmounted and the lessons learned. Through this holistic exploration, we aim to furnish the reader with not only a profound understanding of resource-constrained hardware but also actionable insights into the pragmatic realities of protocol implementation within such exigent contexts.

2. Literature Review

In the realm of the IoT, embedded devices serve as the backbone, requiring both

computational prowess and networking capabilities to fulfill their designated tasks and seamlessly integrate with the vast expanse of the internet. Many times, these Internet of Things devices have low-power embedded computing units included to reduce manufacturing costs. These tools, as outlined in Table 1, embody a spectrum of characteristics tailored to meet the demands of constrained environments [26]. From microcontrollers to System-on-Chip (SoC) solutions, each device offers a unique blend of processing power and energy efficiency, catering to the diverse needs of IoT applications [27] [28]. Through this strategic amalgamation of computational efficiency and networking acumen, IoT devices navigate the intricate web of connectivity, heralding a new era of interconnected intelligence while mitigating the burden of excessive energy consumption and operational costs to furnish the reader with not only a profound understanding of resource-constrained hardware but also actionable insights into the pragmatic realities of protocol implementation within such exigent contexts.

Table 1. Several constrained low-power devices.

Type	CPU	RAM	Flash/ROM
Crossbow TelosB	16-Bit MSP430	10 KB	48 KB
RedBee EconoTAG	32-Bit MC13224v	96 KB	128 KB
Atmel AVR Raven	8-Bit ATMega1284P	16 KB	128 KB
Crossbow Mica2	8-Bit ATMega 128L	4 KB	128 KB

Within the realm of resource-constrained devices like the RedBee EconoTAG, the practical implications of limited storage and memory capacities are vividly illustrated by the findings presented in Table 1. Despite ostensibly providing 96 KB of RAM, the device's operational dynamics unveil a more nuanced reality. A critical examination unveils a notable caveat to this seemingly ample RAM allocation: a substantial portion of this memory is preemptively allocated for pre-execution tasks [29]. Specifically, prior to the commencement of program execution, the contents residing in the flash memory—excluding the bootloader—are mandated to be replicated into RAM. This obligatory pre-execution ritual significantly diminishes the effective availability of RAM for real-time data storage and manipulation [30]. Consequently, the device's nominal memory capacity belies the pragmatic challenges inherent in maximizing its utility within the confines of its constrained hardware ecosystem. This intricate interplay between hardware constraints and software execution protocols underscores the imperative for innovative strategies to optimize resource utilization and enhance operational efficiency in such resource-limited environments.

In designing networking technologies for the IoT, memory constraints are paramount considerations. As such, it is critical to determine a basic set of IP-based protocols capable of effectively managing IoT operations. Equally crucial is the analysis of the bare-minimum feature-set necessary to retain the identity and usability of these protocols with existing tools, ensuring genuine in-

teroperability, while remaining functional on resource-constrained devices [31]. This entails a meticulous balance between functionality and resource efficiency, where protocols must maintain their core characteristics and operational integrity while minimizing memory and processing requirements. By identifying this optimal balance, IoT ecosystems can achieve seamless connectivity and interoperability across diverse devices, enhancing efficiency and scalability in managing IoT networks.

Our goal is to determine the necessary resource requirements for future Internet of Things devices. To this end, we are carrying out a thorough analysis by implementing various IP-based management and security protocols on limited resources. This meticulous approach allows us to discern the minimum requisites necessary for the efficient operation of IoT devices within resource limitations. By implementing various protocols, we aim to evaluate their impact on device performance, ascertain their compatibility with constrained environments, and ultimately identify the optimal balance between functionality and resource consumption. Through this comprehensive analysis, we seek to streamline the deployment of IoT devices, ensuring they remain robust and secure while operating within resource-constrained contexts.

3. Optimized Environment

Selecting the appropriate device and operating system is crucial for implementing and testing management protocols on resource-constrained devices. The chosen operating system must fulfill specific requirements, including providing a 6LoWPAN implementation to facilitate IPv6 connectivity over IEEE 802.15.4 radios. Additionally, it should support both UDP and TCP protocols to enable the implementation of management protocols effectively. By meeting these criteria, the selected OS ensures compatibility with the target network infrastructure and enables seamless communication for managing and monitoring resource-constrained devices efficiently.

When selecting an embedded operating system to meet specific requirements, Contiki emerges as a compelling choice due to its open-source nature, extensive portability, and robust multi-tasking capabilities tailored for embedded devices [32]. Its broad hardware driver support streamlines development efforts and facilitates code portability across various devices. Notably, Contiki's utilization of the µIPv6 stack enables seamless integration with the 6LoWPAN standard, offering advanced networking functionalities. While TinyOS also presents a viable option with its focus on embedded networks and protocol support, Contiki stood out during our implementation phase due to its active developer community and superior IPv6 support [33]. Alternatively, Ethersex caters to simpler networking projects but lacks a mature networking stack compatible with IEEE 802.15.4 standards [26]. Although Embedded Linux offers unparalleled flexibility, its substantial memory requirements, typically around 1 MB, may render it less suitable for resource-constrained environments compared to Contiki and TinyOS [34]-[36].

The Atmel AVR Raven was chosen as the target device especially because Contiki supports it so well, making it perfect for the desired use. Effectively representing resource-constrained network devices, the AVR Raven consists of two primary parts: the RZUSB stick and the Raven boards itself. Connected to a computer, the RZUSB stick acts as a USB Ethernet interface, bridging the IEEE 802.15.4 based network and traditional IP networks. Accessing IEEE 802.15.4 networks is made easy with this interface. Two microcontrollers on the Raven board serve different purposes. The primary microcontroller, the ATMega1284P, handles all processing tasks and interfaces with the AT86RF230 radio for communication. In contrast, the secondary microcontroller, an ATMega3290P, is primarily dedicated to controlling the on-board LCD display. This architecture effectively balances processing and control duties, enabling efficient operation within resource-constrained environments.

A minimalist configuration of Contiki, tailored to operate on the AVR Raven platform while incorporating support for 6LoWPAN, TCP, UDP, and HTTP protocols, efficiently utilizes resources. Its essential functions are given just 6 kB of RAM and 35 kB of ROM, therefore a significant amount of system resources are still unclaimed. More precisely, this configuration makes around 10 kB of RAM and 93 kB of ROM available for further features or apps. Such resource-efficient utilization not only ensures the smooth operation of the Contiki framework but also provides ample room for developers to extend the functionality of the system with custom applications or features, fostering flexibility and scalability in embedded networking environments.

4. Protocol Stack Outline

In navigating the landscape of protocol options for resource-constrained networks, there exists a pivotal choice between embracing innovative, purpose-built protocols like CoAP atop IPv6 infrastructure or leveraging existing protocol sets for seamless interoperability with the established infrastructure [37]. Opting for the latter approach presents distinct advantages, notably the seamless integration of IoT devices into the existing Internet fabric without necessitating substantial modifications. This seamless integration ensures that all incumbent tools remain fully functional and directly applicable to IoT devices, thereby streamlining the adoption process and mitigating potential disruptions to existing workflows. By leveraging the existing protocol ecosystem, IoT deployment can capitalize on the wealth of experience, tools, and infrastructure already in place, fostering a smoother transition and enhancing the overall efficiency and effectiveness of IoT deployments. This strategic alignment with established protocols not only facilitates compatibility but also fosters a robust foundation for scalable, interoperable, and future-proof IoT solutions within the broader Internet landscape.

The selection of a protocol undoubtedly hinges upon the specific needs and constraints of the network in question. Consider, for instance, a deployment scenario characterized by a sparse presence of devices, primarily comprising IoT

devices. In such contexts, opting for one of the latest protocols purpose-built for resource-constrained networks emerges as a strategic choice. These newer protocols, crafted from scratch, offer distinct advantages tailored to the operational demands of such environments. Notably, they are meticulously optimized to accommodate the limitations inherent in resource-constrained networks, thereby ensuring efficiency and reliability in data transmission and communication. By leveraging these protocols, organizations can capitalize on streamlined operations and enhanced performance, thereby fostering seamless connectivity and functionality within their IoT ecosystems as shown in Figure 2.

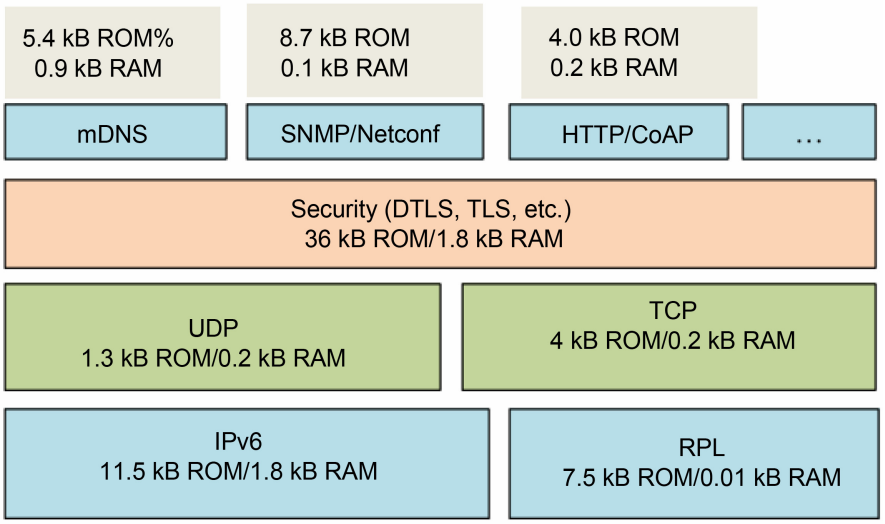

Figure 2. Outline of the protocol stack and detail the memory usage of each individual component on the AVR Raven when integrated with the Contiki OS.

SNMP (Simple Network Management Protocol) and NETCONF (Network Configuration Protocol) are traditional protocols for network management and configuration. SNMP, widely adopted for network monitoring, uses a polling mechanism and a hierarchical namespace but has limitations in security and suitability for constrained devices. NETCONF, using XML for data encoding and secure transport (SSH/TLS), offers fine-grained control and supports configuration validation and rollback, though it is more complex and not tailored for constrained environments. Emerging protocols like CoAP (Constrained Application Protocol), MQTT (Message Queuing Telemetry Transport), and LwM2M (Lightweight Machine-to-Machine) are designed for IoT and constrained devices. CoAP is a RESTful protocol using UDP for lightweight communication, ideal for low power consumption and efficient operation but limited by small payload sizes. MQTT, using a broker-based publish/subscribe model over TCP, is efficient for low-bandwidth, high-latency networks, though reliant on a central broker. LwM2M, based on CoAP, provides comprehensive device management with support for various data formats and remote management capabilities, specifically designed for the IoT ecosystem. Each protocol serves different needs, with SNMP and NETCONF suited for traditional net-

works and CoAP, MQTT, and LwM2M optimized for constrained IoT environments. In scenarios where diverse network devices coexist, leveraging existing network protocols proves pragmatic for meeting management and monitoring demands. Particularly within environments already utilizing IP network infrastructure, deploying established protocols aligns with efficiency and compatibility goals. Considering the growing use of IPv6 in different configurations, the network protocol stack of a generic IoT platform usually incorporates these protocols. While NETCONF conventionally pairs with SSH for security, the protocol's adaptability to other security mechanisms justifies the choice of TLS in this context. Empirical experiments and implementation insights provide the shown ROM and RAM use values, which are supplemented by UDP, TCP, and IPv6 data from earlier publications [38]. This comprehensive approach ensures both reliability and scalability in managing heterogeneous network environments.

5. SNMP on Resource Constrained Devices

SNMP is vital for managing and monitoring network devices. It allows real-time data collection and issue detection, ensuring network health and security. SNMP optimizes performance and supports reliable, efficient network operations, crucial for business continuity. The Contiki SNMP implementation offers robust support for SNMPv1 and SNMPv3 message processing models, catering to diverse network management needs. It enables essential operations like Get, GetNext, and Set, while also facilitating the more efficient GetBulk operation for SNMPv3 [39]. Security is a paramount concern, addressed through the incorporation of the USM security model. This model is bolstered with HMAC-MD5-96 authentication and CFB128-AES-128 symmetric encryption protocols, ensuring data integrity and confidentiality [40]. Despite deviating from the modular architecture defined for SNMP to accommodate device constraints, the implementation maintains reliability and functionality. By optimizing abstract service interfaces and minimizing parameter passing overhead, efficiency is upheld without compromising performance. Important MIB modules, designed especially for the AVR Raven platform, such SNMPv2-MIB, IF-MIB, ENTITY-MIB, and ENTITY-SENSOR-MIB, are painstakingly implemented for thorough testing. This holistic approach underscores Contiki's commitment to delivering a robust, secure, and efficient SNMP solution suitable for diverse network environments.

Compatibility with programs from the Net-SNMP suite and the scli SNMP command line tool has been proven by extensive interoperability testing of the Contiki SNMP implementation [41]. A methodical methodology is used to precisely evaluate the SNMP agent's memory use. First, the operating system's memory footprint without the agent is ascertained. The actual effect of the SNMP agent on system resources is then shown by subtracting this baseline number from the total memory use with the agent activated. Surprisingly, while using little more than 1% of statically assigned RAM, the complete SNMP im-

plementation uses almost 24% of the ROM available on the AVR Raven platform. Interestingly, the agent uses far fewer resources when just SNMPv1 is enabled—roughly 7% of accessible ROM and less than 1% of statically assigned RAM. This meticulous assessment underscores the efficiency and effectiveness of the Contiki SNMP implementation across various configurations and environments.

The analysis of memory usage within the SNMP agent reveals intriguing insights into its composition. Notably, the AES and MD5 implementations command substantial portions of the agent's code size, accounting for approximately 31% and 33%, respectively. Within the AES implementation, a significant portion of ROM is dedicated to storing constants, while the MD5 implementation relies heavily on macro definitions for transformations, contributing to inflated code size. Although optimizing these implementations by replacing macros with functions could potentially reduce code size, such modifications might compromise runtime performance. Notably, there is no optimization for embedded devices in the cryptographic primitives, which comes from the OpenSSL library, suggesting possible inefficiencies. Furthermore, a large chunk of statically allocated RAM is used by the USM security model, mostly for storing the OIDs of error indication counters and localized keys. This intricate interplay between memory allocation, cryptographic functions, and performance considerations underscores the complexity inherent in designing efficient and secure SNMP agents for embedded systems.

Measuring the stack size dynamically poses a challenge, necessitating an experimental approach for estimation. This approach fills a certain bit pattern in the memory section of the program stack when an SNMP message is received. Subsequently, after processing, the stack is analyzed to determine the extent of overwriting. The results, as shown in **Table 2**, depict the maximum stack sizes observed across various SNMP versions and security levels. Remarkably, the answer message buffer takes up the most of the stack—484 bytes in all. Approximately 4% of the RAM is used by the stack size of SNMPv1 and SNMPv3 message processing models without AuthNoPriv security. But when authentication and privacy are turned on, the stack grows by about 66%, to 1144 bytes, or about 7% of the RAM available on the gadget [42] [43].

Table 2. Experimental findings for the highest stack size, measured in bytes, also include the proportion of RAM utilization on an AVR Raven.

Version	Level of Security	Maximum Stack Size
v1	-	688 (4%)
v3	noAuthNoPriv	708 (4%)
v3	authNoPriv	1140 (7%)
v3	authPriv	1144 (7%)

The analysis of SNMP request processing time, as illustrated in **Figure 3**, re-

veals several noteworthy observations. Notably, the duration spent on processing SNMP requests is relatively minimal compared to the time allocated for data transfer. Because SNMPv3's header size is larger in the noAuthNoPriv mode than it is in SNMPv1, latency increases noticeably, especially over low data-rate radio networks. Throughout two protocol variants with basic safety features, the processing delay makes up between 6% and 7% of the total latency. But this statistic is further increased by the addition of identification and privacy measures, rising by around 228% in the worst case. Moreover, it's intriguing to observe that encryption introduces an overhead of roughly 21%, while authentication expenses surge by 58% when contrasted with scenarios devoid of authentication and encryption. These findings align with prior research conducted on more robust devices [42], underscoring the nuanced impact of security measures on SNMP request processing times.

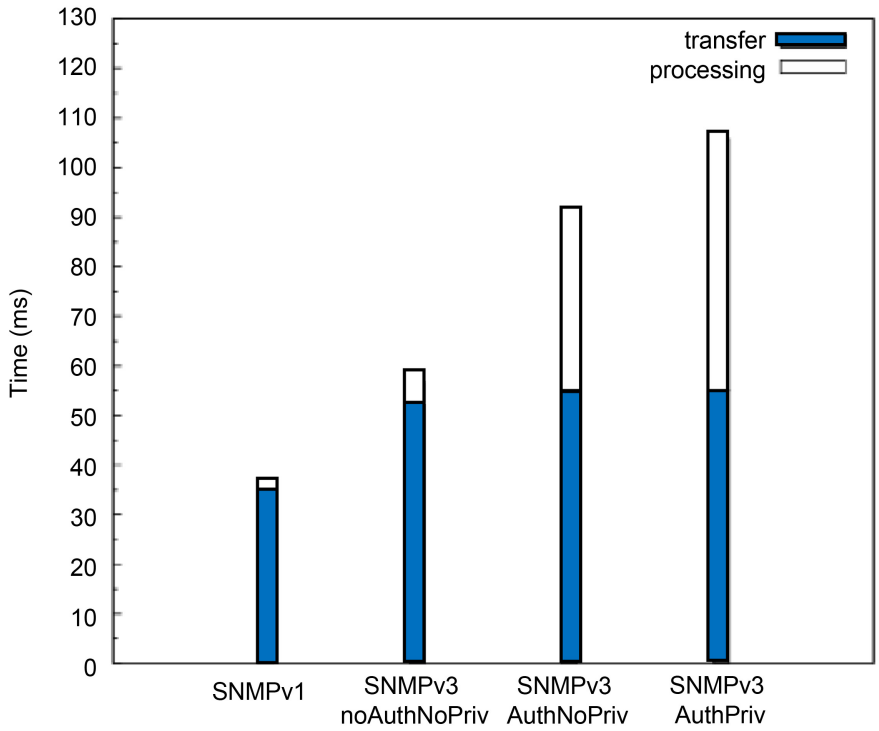

Figure 3. Transfer and execution times for SNMP requests with a single parameter connection averaged.

6. NETCONF on Resource-Constrained Devices

NETCONF enhances network management by standardizing device configuration and automation. It ensures interoperability across various devices and vendors, reduces operational complexity and costs, and supports secure, transactional changes with rollback capabilities. As networks expand with IoT and 5G, NETCONF's role in simplifying and securing management becomes increasingly vital. NETCONF, a network management protocol, facilitates the installation, manipulation, and deletion of configuration settings on network devices. Em-

ploying XML-based encoding for its protocol messages and configuration data, it ensures interoperability and ease of implementation. However, on resource-constrained devices, certain protocol operations, namely get, get-config, copy-config, lock, and unlock, are prioritized due to limitations in processing power and memory. Sub-tree filtering and edit-config operations, though valuable for complex configurations, are omitted in such implementations to streamline functionality and conserve resources. This strategic decision acknowledges the relative simplicity of system configurations on these devices, ensuring efficient utilization of available resources without compromising essential management capabilities.

Our NETCONF implementation's interoperability with current tools is demonstrated by the successful testing of it with the ncclient Python library, which guarantees smooth integration inside network management systems. Our version uses about 25% of the AVR Raven's available ROM, plus an extra 8% of statically allocated RAM, according to our careful testing. Interestingly, these numbers include the contribution made by Contiki's Coffee File System (CFS), which is used to effectively handle NETCONF messages and store configurations, therefore reducing the possibility of RAM overflow. CFS uses around 7% of ROM and just 0.5% of static RAM. Moreover, we developed a dedicated software layer within Contiki to manage configurations and facilitate updates, which incurs a minor overhead of 2% in ROM and 3% in static RAM. These findings not only highlight the resource efficiency of our NETCONF implementation but also showcase the strategic measures taken to optimize memory utilization within the system architecture.

Our NETCONF implementation's efficiency and resource economy are demonstrated by the fact that it uses about 4% of static RAM and about 17% of the available ROM. Remarkably, 6860 bytes in ROM and 170 bytes in static RAM make up the main XML message parsing and generating component of NETCONF. Moreover, using techniques similar to those used in SNMP agent measurements, we investigate stack consumption and find that, like SNMPv1, the NETCONF implementation uses up to 4% of the RAM at most. However, a direct comparison with SNMPv3 is not feasible without incorporating security measures like TLS/DTLS into our NETCONF implementation. This highlights an avenue for potential enhancement in future iterations to align with the robust security standards of SNMPv3.

The processing time for NETCONF requests, as outlined in **Table 3**, notably exceeds that of SNMP, primarily due to the overhead associated with parsing XML-encoded messages and accessing flash memory for reading and writing large messages. While individual read and write operations from flash memory are swift, the generation of XML-encoded messages and writing them to the file accounts for the majority of processing time. Each XML tag necessitates multiple writes, resulting in a significant time investment. For instance, parsing a "get-config" message and generating a response took 0.456 s, largely due to writing the entire configuration tag by tag to the output.xml swap file in flash.

Similarly, the "copy-config" operation took 0.68 s, as it required writing the new configuration to the config.xml file and generating an OK response message in the output.xml. Within the NETCONF implementation, XML processing—which includes actions like "close"—accounted for between 80% and 90% of the overall processing time for significant activities. This implies that using RAM or an XML library designed for less write operations could significantly reduce processing overheads.

Table 3. Average of times for specified network activities.

Operation	TimeTaken
<et-config>	0.592 s
<copy-config>	0.752 s
<close>	0.384 s

With TLS/DTLS for Contiki operating across the 6LoWPAN IPv6 adaption layer, there is hope for protecting communication on limited devices. Even while NETCONF allows running over TLS, on such resource-constrained systems security becomes dependent on using a pre-shared key TLS implementation. Given the similarities between DTLS and TLS, opting for a combined TLS/DTLS implementation presents a pragmatic solution. By integrating this protocol suite into Contiki, we not only fortify data exchanges against unauthorized access and tampering but also align with the evolving security standards in IoT ecosystems. This approach acknowledges the unique challenges posed by constrained environments while prioritizing robust security measures, thereby fostering the safe and efficient operation of interconnected devices within the Internet of Things paradigm.

Conventional RSA or Diffie-Hellman key exchange methods are frequently unworkable in settings limited by bandwidth and processing power. Even if 4 KB RSA is possible to be implemented at Oracle, the sheer volume of messages and processing cycles needed for key creation is still too much. Consequently, a pre-shared key method becomes a pragmatic alternative. AES-CCM is known for being suitable in limited environments, so using AES-128 encryption in the 8-bit Counter and CBC-MAC Mode (CCM) provides a workable options [44]. Interestingly, the IETF CoRE Working Group decided to make this cypher suite mandatory even though it did not completely comply with TLS and DTLS standards since certain mandatory cypher suites were too heavy for limited devices to handle. Nonetheless, our implementations closely adhere to these specifications, albeit without support for session resumption, as doing so would necessitate additional utilization of internal storage resources.

The memory consumption metrics for our TLS and DTLS implementation with PSK-AES-128-CCM-8 are detailed in **Table 4**. Within Contiki, these components dedicated to TLS/DTLS collectively consume about 30% of RAM. Similar to our observations with SNMP, it's evident that cryptographic functions sig-

nificantly contribute to resource utilization. Therefore, transitioning to hardware-supported encryption could substantially mitigate resource overheads, potentially reducing TLS/DTLS resource usage to approximately 23 kB—an approximate 37% improvement. Additionally, the TLS implementation's stack usage peaks at a maximum of 11% of total RAM, while DTLS consumes around 15%. These findings underscore the importance of optimizing cryptographic operations to efficiently manage resource allocation in embedded systems.

Table 4. Items such as memory usage, in bytes, and OTC/DTLS implementations, including percentages on an AVR RAVEN.

Component	RAM	ROM
Contiki mmem*°	516 (3%)	238 (0.2%)
Contiki CFS*	92 (0.5%)	7502 (6%)
AES-CCM*°	310 (2%)	14,058 (11%)
HMAC-SHA256*°	288 (2%)	3594 (3%)
TLS*	655 (4%)	12,048 (9%)
DTLS°	847 (5%)	19,342 (15%)
TLSTotal	1861 (11%)	37,440 (29%)
DTLSTotal	1961 (12%)	37,232 (28%)

Components marked with* are used by TIS and ° are used by DTLS.

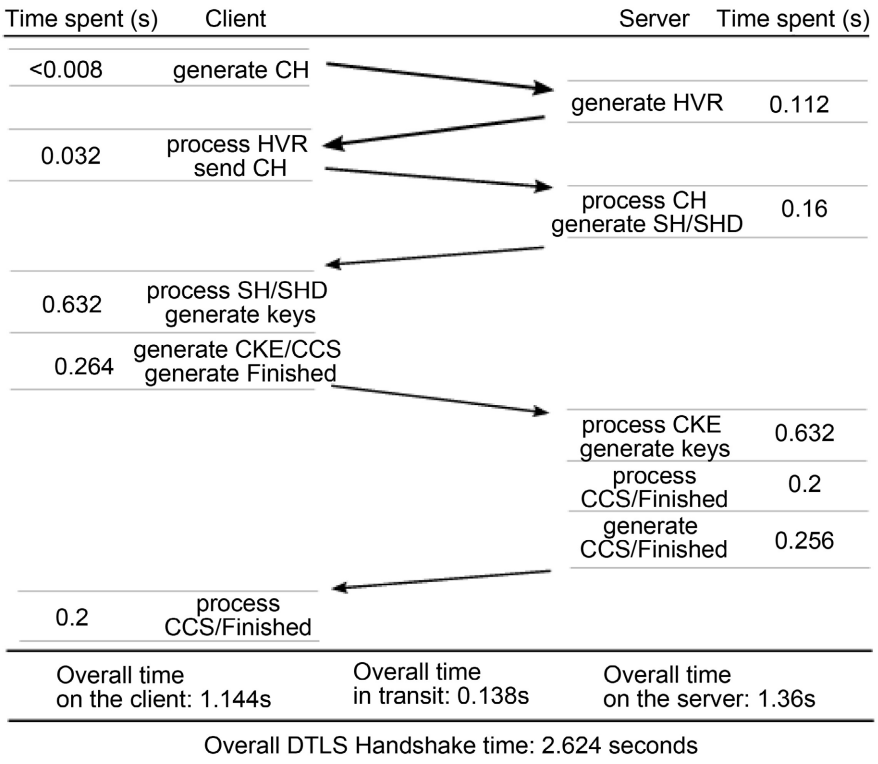

Figure 4. Explanation of how long a DTLS handshake takes. (CH-Client Hello; HVR-Hello Verify Request; SH-Server Hello; SHD Server Hello Done; CKE-Client Key Exchange; CCS-Change Cipher Spec).

The bulk of time in the DTLS handshake is spent on key generation and producing the finished messages, as seen in Figure 4. The many HMAC-SHA256 calculations required to make these computing chores time-consuming. Hashing increases the overhead as well, especially when creating the Hello Verify Request (HVR), in which the cookie delivered to the client is a hash of several fields from the Client Hello (CH) message. While not stated in detail in the breakdown, DTLS and the TLS handshake include comparable laborious procedures. There are differences, though: a TCP connection needs to be made, which takes about 0.048 seconds, before the handshake can be started. In addition, TLS saves the client and server 0.1 and 0.032 seconds, respectively, by having the server skip delivering the HVR. Furthermore, the TLS handshake involves fewer messages sent, which reduces the time spent in transit. Overall, the TLS handshake concludes within an average time of 2.496 seconds.

7. Conclusions

In our investigation, we delved into the feasibility of managing constrained networks and devices within the Internet of Things (IoT) framework by leveraging existing network management protocols. We obtained important understanding of their resource requirements by using the Contiki operating system to construct condensed versions of SNMP and NETCONF together with the security mechanisms that go along with them. Implementing NETCONF and SNMP on 8-bit AVR-based devices is challenging due to their limited resources. These protocols are crucial for secure network management, but the constrained processing power, memory, and energy of these microcontrollers require highly efficient implementations. In real-world IoT applications, such as smart grids and industrial control systems, ensuring robust security without overwhelming the device is essential. Failures in these systems can cause significant disruptions and financial losses, making the balance between security and resource efficiency critical for reliable and secure IoT deployments. Our endeavor aims to furnish a comprehensive set of data points, serving as a foundational reference for future evaluations against alternative methodologies for managing constrained networks and IoT devices. Central to our findings are the pivotal challenges we encountered, particularly concerning message framing intricacies, the establishment and upkeep of sessions, and the critical matter of security. Addressing these challenges entails circumventing issues such as message reassembly across multiple layers, ensuring efficient mechanisms for applying configuration changes, and grappling with the resource-intensive nature of session key generation, even with pre-shared keys under TLS and DTLS protocols.

The comparison of SNMP and NETCONF implementations highlights the efficiency of SNMP, particularly on resource-constrained devices, with simple requests processed within 40 - 120 ms. In contrast, NETCONF requires notably more time, ranging from 500 - 900 ms for basic requests, excluding security considerations. However, potential optimizations, such as minimizing flash memory

access for short messages, could enhance NETCONF's performance. Yet, adding security measures, especially with TLS, significantly extends session start-up times, though hardware encryption support could alleviate this burden. While SNMP's USM security mechanism relies on pre-shared keys without generating session keys, TLS/DTLS involves more complex processes, with elliptic curve cryptography proving feasible even on constrained devices, as demonstrated by the ability to complete a TLS handshake in less than 4 seconds. Despite room for further optimization, the slight disparity of just over 1 second compared to pre-shared key setups underscores the surprising viability of elliptic curve cryptography in this context.

The proposal for a programmatic RESTful interface as an alternative to the NETCONF protocol, particularly for resource-constrained devices, marks a significant shift towards efficiency and scalability. By leveraging CoAP's compact encoding and support for JSON data, it offers a more lightweight solution, mitigating the overhead associated with TCP sessions and reducing message size. This approach not only addresses memory constraints but also simplifies implementation by eliminating the need for complex session management. Moreover, the decoupling of DTLS security associations from TCP sessions maintains security integrity while potentially improving performance. As this proposal advances, practical implementation experience will be crucial in validating its efficacy and assessing its impact on device management and operation context.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Sovacool, B.K. and Furszyfer Del Rio, D.D. (2020) Smart Home Technologies in Europe: A Critical Review of Concepts, Benefits, Risks and Policies. *Renewable and Sustainable Energy Reviews*, **120**, Article 109663. https://doi.org/10.1016/j.rser.2019.109663

[2] Al Mahmud, M.A., Hossain, M.A., Saju, M.A.B., Ul-lah, M.W., Hasan, R. and Suzer, G. (2024) Information Technology for the Next Future World: Adoption of It for Social and Economic Growth: Part II. *International Journal of Innovative Research in Technology*, **10**, 742-747.

[3] Mohammad, N., Imran, M.A.U., Prabha, M., Sharmin, S. and Khatoon, R. (2024) Combating Banking Fraud with It: Integrating Machine Learning and Data Analytics. *The American Journal of Management and Economics Innovations*, **6**, 39-56. https://doi.org/10.37547/tajmei/volume06issue07-04

[4] Hasan, R., Farabi, S.F., Al Mahmud, M.A., Akter, J. and Hossain, M.A. (2024) Information Technologies for the Next Future World: Implications, Impacts and Barriers: Part I. *International Journal of Creative Research Thoughts*, **12**, a323-a330.

[5] Shahana, A., Hasan, R., Farabi, S.F., Akter, J., Mahmud, M.A.A., Johora, F.T., *et al*. (2024) Ai-Driven Cybersecurity: Balancing Advancements and Safeguards. *Journal of Computer Science and Technology Studies*, **6**, 76-85.

https://doi.org/10.32996/jcsts.2024.6.2.9

[6] Ahmed, E. and Sobuz, H.R. (2011) Flexural and Time-Dependent Performance of Palm Shell Aggregate Concrete Beam. *KSCE Journal of Civil Engineering*, **15**, 859-865. https://doi.org/10.1007/s12205-011-1148-2

[7] Akid, A.S.M., Shah, S.M.A., Sobuz, M.D.H.R., Tam, V.W.Y. and Anik, S.H. (2021) Combined Influence of Waste Steel Fibre and Fly Ash on Rheological and Mechanical Performance of Fibre-Reinforced Concrete. *Australian Journal of Civil Engineering*, **19**, 208-224. https://doi.org/10.1080/14488353.2020.1857927

[8] Akid, A.S.M., Wasiew, Q.A., Sobuz, M.H.R., Rahman, T. and Tam, V.W. (2020) Flexural Behavior of Corroded Reinforced Concrete Beam Strengthened with Jute Fiber Reinforced Polymer. *Advances in Structural Engineering*, **24**, 1269-1282. https://doi.org/10.1177/1369433220974783

[9] Li, Z., He, X., Ding, Z., Hossain, M.M., Rifat, M.T.R., Sobuz, M.H.R., *et al*. (2024) Analysis of Influencing Factors for Housing Construction Technology in Desakota Village and Town Communities in China. *Humanities and Social Sciences Communications*, **11**, Article No. 432. https://doi.org/10.1057/s41599-024-02937-2

[10] Rahman Sobuz, M.H., Alam, A., John Oehlers, D., Visintin, P., Hamid Sheikh, A., Mohamed Ali, M.S., *et al*. (2023) Experimental and Analytical Studies of Size Effects on Compressive Ductility Response of Ultra-High-Performance Fiber-Reinforced Concrete. *Construction and Building Materials*, **409**, Article 133864. https://doi.org/10.1016/j.conbuildmat.2023.133864

[11] Abir, S.M.A.A., Anwar, A., Choi, J. and Kayes, A.S.M. (2021) Iot-Enabled Smart Energy Grid: Applications and Challenges. *IEEE Access*, **9**, 50961-50981. https://doi.org/10.1109/access.2021.3067331

[12] Shi, X., An, X., Zhao, Q., Liu, H., Xia, L., Sun, X., *et al*. (2019) State-of-the-Art Internet of Things in Protected Agriculture. *Sensors*, **19**, Article 1833. https://doi.org/10.3390/s19081833

[13] Vermesan, O., Friess, P., Guillemin, P., Giaffreda, R., Grindvoll, H., Eisenhauer, M., *et al*. (2022) Internet of Things Beyond the Hype: Research, Innovation and Deployment. In: Vermesan, O. and Friess, P., Eds., *Building the Hyperconnected Society—Internet of Things Research and Innovation Value Chains, Ecosystems and Markets*, River Publishers, 15-118. https://doi.org/10.1201/9781003337454-3

[14] Hasan, R., Al Mahmud, M.A., Farabi, S.F., Akter, J. and Johora, F.T. (2024) Unsheltered: Navigating California's Homelessness Crisis. *Sociology Study*, **14**, 143-156. https://doi.org/10.17265/2159-5526/2024.03.002

[15] Hasan, R., Chy, M.A.R., Johora, F.T., Ullah, M.W. and Saju, M.A.B. (2024) Driving Growth: The Integral Role of Small Businesses in the U.S. Economic Landscape. *American Journal of Industrial and Business Management*, **14**, 852-868. https://doi.org/10.4236/ajibm.2024.146043

[16] Hasan, R., Farabi, S.F., Kamruzzaman, M., BHUYAN, M.K., Nilima, S.I. and Shahana, A. (2024) Ai-Driven Strategies for Reducing Deforestation. *The American Journal of Engineering and Technology*, **6**, 6-20. https://doi.org/10.37547/tajet/volume06issue06-02

[17] Johora, F.T., Hasan, R., Farabi, S.F., Akter, J. and Mahmud, M.A.A. (2024) AI-Powered Fraud Detection in Banking: Safeguarding Financial Transactions. *The American Journal of Management and Economics Innovations*, **6**, 8-22. https://doi.org/10.37547/tajmei/volume06issue06-02

[18] Montenegro, G., Kushalnagar, N., Hui, J. and Culler, D. (2007) Transmission of IPv6 Pack-Ets over IEEE 802.15. 4 Networks.

[19] Saavedra, E., Santamaria, A., del Campo, G. and Gomez, I. (2024) Leveraging IoT Harmonization: An Efficacious Nb-IoT Relay for Integrating 6Lowpan Devices into Legacy Ipv4 Networks. *Applied Sciences*, **14**, Article 3411. https://doi.org/10.3390/app14083411

[20] Triantafyllou, A., Sarigiannidis, P. and Lagkas, T.D. (2018) Network Protocols, Schemes, and Mechanisms for Internet of Things (IoT): Features, Open Challenges, and Trends. *Wireless Communications and Mobile Computing*, **2018**, Article ID: 5349894. https://doi.org/10.1155/2018/5349894

[21] Habibur Rahman Sobuz, M., Khan, M.H., Kawsarul Islam Kabbo, M., Alhamami, A.H., Aditto, F.S., Saziduzzaman Sajib, M., *et al*. (2024) Assessment of Mechanical Properties with Machine Learning Modeling and Durability, and Microstructural Characteristics of a Biochar-Cement Mortar Composite. *Construction and Building Materials*, **411**, Article 134281. https://doi.org/10.1016/j.conbuildmat.2023.134281

[22] Harrington, D., Presuhn, R. and Wijnen, B. (2002) An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks.

[23] Enns, R. (2006) NETCONF Configuration Protocol.

[24] Schonwalder, J., Bjorklund, M. and Shafer, P. (2010) Network Configuration Management Using NETCONF and Yang. *IEEE Communications Magazine*, **48**, 166-173. https://doi.org/10.1109/mcom.2010.5560601

[25] Deep, S., Zheng, X., Jolfaei, A., Yu, D., Ostovari, P. and Kashif Bashir, A. (2020) A Survey of Security and Privacy Issues in the Internet of Things from the Layered Context. *Transactions on Emerging Telecommunications Technologies*, **33**, e3935. https://doi.org/10.1002/ett.3935

[26] Sehgal, A., Perelman, V., Kuryla, S. and Schonwalder, J. (2012) Management of Resource Constrained Devices in the Internet of Things. *IEEE Communications Magazine*, **50**, 144-149. https://doi.org/10.1109/mcom.2012.6384464

[27] Chakravarthi, V.S. (2020) A Practical Approach to VLSI System on Chip (SoC) Design. Springer.

[28] Chakravarthi, V.S. and Koteshwar, S.R. (2023) System on Chip (SOC) Architecture: A Practical Approach. Springer Nature.

[29] Cavicchio, J.C. (2019) Effective and Efficient Preemption Placement for Cache Over-Head Minimization in Hard Real-Time Systems.

[30] Vermesan, O. and Friess, P. (2013) Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems. River Publishers.

[31] Zakurdaev, G.M. (2023) A Scalable Approach to Improve Security and Resilience of Smart City IoT Architectures.

[32] Amjad, M., Sharif, M., Afzal, M.K. and Kim, S.W. (2016) Tinyos-New Trends, Comparative Views, and Supported Sensing Applications: A Review. *IEEE Sensors Journal*, **16**, 2865-2889. https://doi.org/10.1109/jsen.2016.2519924

[33] Wang, A., Zha, Z., Guo, Y. and Chen, S. (2019) Software-Defined Networking Enhanced Edge Computing: A Network-Centric Survey. *Proceedings of the IEEE*, **107**, 1500-1519. https://doi.org/10.1109/jproc.2019.2924377

[34] Looga, V. (2018) Energy Efficiency in Large-Scale Internet of Things Networks.

[35] Jabin, J.A., Khondoker, M.T.H., Sobuz, M.H.R. and Aditto, F.S. (2024) High-Temperature Effect on the Mechanical Behavior of Recycled Fiber-Reinforced Concrete Containing Volcanic Pumice Powder: An Experimental Assessment Combined with Machine Learning (ML)-Based Prediction. *Construction and Building Materials*, **418**, Article 135362.

https://doi.org/10.1016/j.conbuildmat.2024.135362

[36] Sobuz, M.H.R., Joy, L.P., Akid, A.S.M., Aditto, F.S., Jabin, J.A., Hasan, N.M.S., *et al*. (2024) Optimization of Recycled Rubber Self-Compacting Concrete: Experimental Findings and Machine Learning-Based Evaluation. *Heliyon*, **10**, e27793. https://doi.org/10.1016/j.heliyon.2024.e27793

[37] Shelby, Z., Hartke, K., Bormann, C. and Frank, B. (2012) Constrained Application Protocol (CoAP). Draft-Ietfcore-Coap-12.

[38] Ekpenyong, M.E., Asuquo, D.E., Udo, I.J., Robinson, S.A. and Ijebu, F.F. (2022) Ipv6 Routing Protocol Enhancements over Low-Power and Lossy Networks for IoT Applications: A Systematic Review. *New Review of Information Networking*, **27**, 30-68. https://doi.org/10.1080/13614576.2022.2078396

[39] Marwaha, M. (2021) Model-Driven Device Provisioning.

[40] Perelman, V. and Ersue, M. (2012) TLS with PSK for Constrained Devices.

[41] Savic, M. (2016) Bridging the SNMP Gap: Simple Network Monitoring the Internet of Things. *Facta Universitatis Series*: *Electronics and Energetics*, **29**, 475-487. https://doi.org/10.2298/fuee1603475s

[42] Gondim, J.J.C., de Oliveira Albuquerque, R. and Sandoval Orozco, A.L. (2020) Mirror Saturation in Amplified Reflection Distributed Denial of Service: A Case of Study Using SNMP, SSDP, NTP and DNS Protocols. *Future Generation Computer Systems*, **108**, 68-81. https://doi.org/10.1016/j.future.2020.01.024

[43] Sobuz, M.H.R., Al-Imran,, Datta, S.D., Jabin, J.A., Aditto, F.S., Sadiqul Hasan, N.M., *et al*. (2024) Assessing the Influence of Sugarcane Bagasse Ash for the Production of Eco-Friendly Concrete: Experimental and Machine Learning Approaches. *Case Studies in Construction Materials*, **20**, e02839. https://doi.org/10.1016/j.cscm.2023.e02839

[44] McGrew, D. and Bailey, D. (2012) Aes-ccm Cipher Suites for Transport Layer Security (TLS).