Contents

Preface vii

1. A comprehensive survey of issues in solid state drives 1
   Youngbin Jin and Ben Lee
   1. Introduction 2
   2. Fundamentals of a flash device 4
   3. SSD architecture and flash memory structure 9
   4. Flash translation layer 15
   5. Address mapping 16
   6. Garbage collection 29
   7. Wear leveling 32
   8. Bad block management 34
   9. SSD parallelism techniques 35
   10. Page allocation 39
   11. Recent research trend 41
   12. Conclusion 64
   References 65
   About the authors 68

2. Revisiting VM performance and optimization challenges for big data 71
   Muhammad Ziad Nayyer, Imran Raza, and Syed Asad Hussain
   1. Introduction 72
   2. Performance issues from big data perspective 74
   3. Migration overhead mitigation techniques from big data perspective 83
   4. Summary, insights and open research challenges 98
   5. Conclusion 105
   References 106
   About the authors 111

3. Toward realizing self-protecting healthcare information systems: Design and security challenges 113
   Qian Chen
   1. Introduction 115
   2. Autonomic computing and self-protecting HISs 120
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. Healthcare information system</td>
<td>126</td>
</tr>
<tr>
<td>4. HL7 messaging standard</td>
<td>128</td>
</tr>
<tr>
<td>5. State-of-the-art research for enhancing HIS security</td>
<td>133</td>
</tr>
<tr>
<td>6. Conclusion</td>
<td>142</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>142</td>
</tr>
<tr>
<td>References</td>
<td>143</td>
</tr>
<tr>
<td>About the author</td>
<td>149</td>
</tr>
<tr>
<td><strong>4. SSIM and ML based QoE enhancement approach in SDN context</strong></td>
<td>151</td>
</tr>
<tr>
<td>Asma Ben Letaifa</td>
<td></td>
</tr>
<tr>
<td>1. Introduction and challenges</td>
<td>153</td>
</tr>
<tr>
<td>2. About QoS/QoE parameters</td>
<td>155</td>
</tr>
<tr>
<td>3. Environment and challenges</td>
<td>159</td>
</tr>
<tr>
<td>4. QoS and QoE metrics in different services applications</td>
<td>165</td>
</tr>
<tr>
<td>5. How modeling QoS/QoE in related work?</td>
<td>166</td>
</tr>
<tr>
<td>6. Video streaming protocols</td>
<td>168</td>
</tr>
<tr>
<td>7. Enhancing QoE approaches</td>
<td>175</td>
</tr>
<tr>
<td>8. Simulation environment and testbed</td>
<td>178</td>
</tr>
<tr>
<td>9. Proposed approach: A ML-based enhancing QoE approach</td>
<td>181</td>
</tr>
<tr>
<td>10. Conclusion, perspectives and futures work</td>
<td>190</td>
</tr>
<tr>
<td>References</td>
<td>190</td>
</tr>
<tr>
<td>About the author</td>
<td>196</td>
</tr>
</tbody>
</table>
Traditionally, *Advances in Computers*, the oldest series to chronicle the rapid evolution of computing, annually publishes several volumes, each one typically comprised of four to eight chapters, describing new developments in the theory and applications of computing. The 114th volume is an eclectic volume inspired by several issues of interest in research and development in computer science. This volume is a collection of four chapters that were solicited from authorities in the field, each of whom brings to bear a unique perspective on the topic.

Due to its high performance and density flash memory-based solid-state drives (SSDs) are becoming ubiquitous in modern computing systems. And, they are widely used as a secondary memory because of their superior performance in comparison to magnetic hard disk drives. In Chapter 1, “A comprehensive survey of issues in solid state drives,” Youngbin Jin and Ben Lee present a comprehensive study of SSDs topics that span across from the physical properties of a flash memory cell to the architecture of SSDs. In addition, it addresses recent research efforts on SSDs.

In Chapter 2, “Revisiting VM performance & optimization challenges for big data,” Nayyer et al. articulate the concept of virtualization in cloud computing as a means to maximize resource utilization and minimize cost by deploying multiple virtual machines on a single physical platform. However, sharing computing resources by multiple virtual machines can result in severe performance degradation, motivating virtual machine migration techniques. Introduction of big data brings out new challenges to the existing virtual machines performance enhancement. This chapter describes big data-triggered VM performance challenges focusing on big data applications and storage migration in cloud computing. State-of-the-art VM migration techniques are evaluated against challenges posed by big data to motivate possible solutions and research challenges.

In Chapter 3 entitled, “Toward realizing self-protecting healthcare information systems: Design and security challenges,” Qian Chen studies the history of Healthcare Information Systems (HISs) and discusses recent cyber security threats affecting HISs and then introduces the autonomic computing concept. This chapter also analyzes the security vulnerabilities of the HIS network, communication links, and protocols. Based on these studies the concept of self-protecting HISs (SPHISs) that can defend
themselves against cyber intrusions with little or no human intervention is articulated. This chapter articulates that a SPHIS should contain monitoring systems, early estimation modules, intrusion detection, network forensics analysis devices, and intrusion response systems. Finally, existing self-protecting approaches for HIS, enterprise systems, and industrial control systems are demonstrated in detail.

Finally, in Chapter 4, “SSIM and ML based QoE enhancement approach in SDN context,” Asma Ben Letaifa is looking at a new performance metric, i.e., quality of experience within the framework of video streaming. This chapter first articulates the so-called QoS, QoE, mobile cloud computing environment, and software-defined network. Mathematical techniques in modeling, predicting, and evaluating QoE are presented. Finally, a machine learning adaptive coding to provide a better QoE for video streaming services is presented and evaluated.

I hope that readers find these articles of interest, and useful for teaching, research, and other professional activities. I welcome feedback on this volume, as well as suggestions for topics of future volumes.

Ali R. Hurson
Missouri University of Science and Technology, Rolla, MO, United States
CHAPTER ONE

A comprehensive survey of issues in solid state drives

Youngbin Jin, Ben Lee
School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR, United States

Contents

1. Introduction 2
2. Fundamentals of a flash device 4
   2.1 Flash device 4
   2.2 Basic operations of flash memory 5
   2.3 Endurance 8
3. SSD architecture and flash memory structure 9
   3.1 SSD architecture 9
   3.2 Structure of a flash device 12
4. Flash translation layer 15
5. Address mapping 16
   5.1 Page-level mapping 17
   5.2 Block-level mapping 17
   5.3 Hybrid mapping 18
6. Garbage collection 29
   6.1 FIFO GC algorithm 30
   6.2 Greedy GC algorithm 31
   6.3 Windowed GC algorithm 31
   6.4 d-Choice GC algorithm 32
7. Wear leveling 32
   7.1 Evenness-aware algorithm 33
   7.2 Dual-pool algorithm 33
8. Bad block management 34
   8.1 Bad block recognition 34
   8.2 Bad block replacement 34
9. SSD parallelism techniques 35
   9.1 Channel striping 36
   9.2 Flash-chip pipelining 36
   9.3 Die interleaving 36
   9.4 Plane sharing 38
   9.5 Combining parallelism techniques 38
Abstract

Flash memory-based solid state drives (SSDs) have become ubiquitous in modern computing systems, such as high-performance servers, workstation, desktops, and laptops, due to their performance and density. The architecture of SSDs has evolved to exploit the advantages of flash memories and at the same time conceal their shortcomings. SSD concurrency techniques such as channel striping, flash-chip pipelining, die interleaving, and plane sharing utilize the available parallelism of flash memories, while the flash translation layer (FTL) operations minimize the latency overhead of flash memories. This chapter provides a comprehensive survey of SSD topics that span across from the physical properties of a flash memory cell to the architecture of SSDs. The FTL-related topics are discussed in the context of inter-related system-level operations, which include address mapping, garbage collection, wear leveling, bad block management, SSD concurrency techniques, and page allocation strategies. This chapter also surveys recent research efforts on SSDs.

1. Introduction

In the recent years, flash memories have become the main storage technology for computers and mobile devices. NAND flash memory-based solid-state drives (SSDs) are widely used as a secondary memory in modern computing systems due to their superior performance compared to magnetic hard disk drives (HDDs). The performance of HDDs has stagnated due to the limitations in the rotational speed of magnetic platters and the seek time of actuator arms. In contrast, SSDs do not have complex mechanical parts, resulting in lower latency as well as lower failure rate than HDDs [1]. Furthermore, SSDs offer superior bandwidth, higher random I/O performance, lower power consumption, higher shock resistance, and improved system reliability compared to HDDs [2, 3].

The first version of SSDs were RAM-based invented by StorageTeK in 1978. Flash memory-based SSDs were introduced in 1989 by Western Digital. Early SSDs were implemented with the NOR flash memory structure due to its high performance. SSDs with the higher density NAND flash
memory structure were developed by M-System in 1995; however, their use was limited to certain applications (e.g., in industrial or military settings) due to their high cost. By 2004, the cost of NAND flash memories dropped drastically, and SSDs entered a new phase as a replacement for magnetic storage devices. Flash memories have since been used as storage for portable devices, such as smartphones, pad devices, laptops, etc. They are also used in desktops and high-performance servers due to their high capacity and bandwidth [2, 4].

Despite the popularity and advantages of SSDs, flash memories have critical shortcomings. First, a flash memory cell has to be erased first before it can be programmed, referred to as erase-before-write, due to its physical characteristics. Furthermore, the erase operation has a long latency that needs to be hidden. Second, the oxide layer in a flash memory cell will become damaged over many program and erase (P/E) cycles. This causes the flash memory cell to become unreliable limiting its lifetime, which is referred to as the endurance problem [3].

SSD architectures have been developed to exploit the advantages of flash memories and to hide their disadvantages. In order to achieve these goals, SSDs employ flash translation layer (FTL) to perform various operations to maximize their performance, lifetime, and reliability. FTL consists of the following functionalities:

- Maps logical addresses from the host system to physical addresses of flash memories.
- Due to the erase-before-write requirement of flash memory, write requests to valid pages (i.e., updates) must be performed on new blocks and the original pages need to be marked as invalid. This process called out-of-place updating increases the number of invalid pages. Therefore, these invalid pages need to be reclaimed for future allocation by performing a background operation referred to as garbage collection.
- Since flash memories have a limited number of P/E cycles, their reliability is not guaranteed when the number of P/E cycles exceeds the maximum threshold. Therefore, these P/E cycles need to be managed by performing wear leveling to evenly wear out flash blocks [1–3, 5].
- Performs bad block management to avoid using blocks that have exceeded the P/E cycle limit and thereby maintain high reliability [6, 7].
- Exploits concurrency using techniques such as channel striping, flash-chip pipelining, die interleaving, and plane sharing in order to maximize performance.
- Checks the reliability of blocks using error correction code (ECC).
Based on the aforementioned discussion, this chapter provides a comprehensive survey of issues in SSDs. Unlike existing surveys that are limited to specific areas of SSD, such as FTL [8, 9], endurance and reliability [10], or architecture [11], this chapter provides a broad coverage of issues from the Flash device-level to the SSD system-level. Moreover, this chapter also surveys recent research efforts on SSDs.

The chapter is organized as follows: Section 2 discusses the fundamentals of a flash device, such as material composition, the flash memory characteristics that allow “0” and “1” to be distinguished, and inherent endurance limitation. Based on this background, the rest of the section covers the basic operations of a NAND flash memory. Section 3 provides a brief description of the SSD architecture and discusses the structure of a flash memory from one transistor cell to an entire SSD device. The importance of FTL is introduced in Section 4. Afterward, Sections 5–8 cover address mapping, garbage collection, wear leveling, and bad block management, respectively. Section 9 discusses the various SSD parallelism techniques. Section 10 introduces page allocation strategies on SSDs. Recent research trends in SSDs are presented in Section 11. Finally, Section 12 concludes the chapter.

2. Fundamentals of a flash device

SSDs are designed to exploit the advantages of flash devices and at the same time to conceal their disadvantages. For this reason, this section presents the fundamentals of flash memory, including the device structure and basic operations, to better understand flash memory-based SSDs.

2.1 Flash device

A flash memory cell is made up of a metal oxide semiconductor field effect transistor (MOSFET) with a floating gate as shown in Fig. 1. A floating gate is inserted between the metal control gate and the substrate layer. This floating gate allows the flash cell to maintain one of the two possible states, which are distinguished by whether or not there are electrons trapped in the floating gate.

Fig. 2 illustrates the states “1” and “0”. When there are no electrons in the floating gate, the state of the cell is considered to be “1”. When there are electrons in the floating gate, the state of the cell is considered to be “0”. The trapped electrons in the floating gate shift the threshold voltage ($V_t$) level of the MOSFET. Thus, a floating gate with trapped electrons has a higher $V_t$
than a floating gate with no electrons, i.e., \( V_{\text{data}}^1 < V_{\text{data}}^0 \). In order to read the state of a cell, a read reference voltage \( (V_{\text{READ}}) \) is applied to the control gate, where \( V_{\text{data}}^1 < V_{\text{READ}} < V_{\text{data}}^0 \) [12]. Then, a sensing circuit distinguishes whether or not there is current flowing through the cell to read data. If \( V_t \) is higher than \( V_{\text{READ}} \), there will be no current through the cell. On the other hand, if \( V_t \) is lower than \( V_{\text{READ}} \), current passes through the cell. Note that the floating gate can retain electrons for a few years without power, and this physical characteristic (i.e., long retention time) allows flash memories to be used as nonvolatile storage [12, 13].

2.2 Basic operations of flash memory

The basic operations of flash memory are program, read, and erase. Fig. 3 illustrates the program and erase operations. The following explains the three operations in more detail:

2.2.1 Program

A program operation involves injecting electrons into the floating gate to write a “0”. The Fowler–Nordheim (FN) tunneling is used to inject
electrons into the floating gate. FN tunneling occurs when a high voltage (e.g., 18 V) is applied across the control gate and the substrate as shown in Fig. 3A [14].

2.2.2 Read
As discussed in Section 2.1, the basic principle of a read operation is based on \(V_t\). Fig. 4 shows the \(V_t\) distribution of a single-level cell (SLC) flash memory structure that stores one bit of data, where \(V_t\) for data “1” (\(V_{t^{data1}}\)) is lower than \(V_{READ}\), while \(V_t\) for data “0” (\(V_{t^{data0}}\)) is higher than \(V_{READ}\). \(V_{PASS}\) is a voltage level higher than any of the data states to pass the current during a page read operation (see Section 3.2). In order to perform a read operation, \(V_{READ}\) is applied to the control gate of a flash cell being read. Then, the stored data can be determined based on whether or not current passes through the transistor. That is, if current is detected, it means the cell contains data “1”. Otherwise, the cell contains data “0”.

Fig. 5 shows the \(V_t\) distribution of a multilevel cell (MLC) flash memory technology, which is divided into four voltage windows to store two bits of data in one flash cell. A read operation involves a two-step process. First, the most significant bit is detected by applying \(V_{READ2}\) to distinguish between data sets “1x” and “0x”. Afterward, the least significant bit can be detected by applying \(V_{READ1}\) to distinguish between data “11” and “10” or \(V_{READ3}\) to distinguish between data “01” and “00”. Note that a triple-level cell (TLC) flash memory technology can hold three bits of data in one flash cell.
2.2.3 Erase

An erase operation is the process of removing electrons from the floating gate in order to change the state of the cell to “1” [12, 15]. During an erase operation, a large negative voltage is required to repel electrons from the floating gate as shown in Fig. 3B [12, 15]. This is accomplished by grounding the control gate and applying 20 V to the substrate. As a result, electrons are removed from the floating gate due to the FN tunneling effect.

The erase operation is performed on a block-by-block basis, which means that an individual flash cell cannot be changed from “0” to “1”, unlike from “1” to “0” as shown in Fig. 6. Thus, new empty (erased) blocks have to be prepared before programming (i.e., erase-before-write). Moreover, the erase operation has a much longer latency than the read and program operations. For example, the read, program, and erase latencies for Micron 8 Gb flash-chip are 25 μs, 220 μs, and 1500 μs, respectively [16]. For this reason, the erase operation is one of the critical performance bottlenecks for NAND flash memories, and many FTL algorithms are designed to conceal the long erase latency (Fig. 6).

Note that it is possible to apply –20 V at the control gate. However, this requires an extra charge pump, which adds cost in terms of power consumption and hardware implementation.
2.3 Endurance

As mentioned in Section 1, a flash cell has a limited number of P/E cycles. Typical P/E cycles for SLC, MLC, and TLC are 100 k, 10 k, and 1 k, respectively [17]. If these limitations are exceeded, their reliability as flash memories is not guaranteed. This endurance problem is caused by the damaged oxide layer due to repeated P/E operations as shown in Fig. 7 [10, 18–20]. The damaged oxide layer causes (1) electrons to become trapped in the tunnel oxide layer increasing \( V_t \) for an erased cell and (2) electrons to escape from the floating gate faster due to stress-induced leakage current (SILC) reducing retention time of a programmed cell. Fig. 8 shows how \( V_t \) of an erased cell increases as P/E cycles increase [10]. After \( V_t \) crosses the reference voltage \( V_{READ} \), the cell cannot reliably operate as a memory element.

![Comparison of a normal cell (A) and a damaged cell (B) due to repeated P/E operations.](image)

![SLC flash cell endurance characteristics.](image)

**Fig. 7** Comparison of a normal cell (A) and a damaged cell (B) due to repeated P/E operations.

**Fig. 8** SLC flash cell endurance characteristics [10].

2.3 Endurance

As mentioned in Section 1, a flash cell has a limited number of P/E cycles. Typical P/E cycles for SLC, MLC, and TLC are 100 k, 10 k, and 1 k, respectively [17]. If these limitations are exceeded, their reliability as flash memories is not guaranteed. This endurance problem is caused by the damaged oxide layer due to repeated P/E operations as shown in Fig. 7 [10, 18–20]. The damaged oxide layer causes (1) electrons to become trapped in the tunnel oxide layer increasing \( V_t \) for an erased cell and (2) electrons to escape from the floating gate faster due to stress-induced leakage current (SILC) reducing retention time of a programmed cell. Fig. 8 shows how \( V_t \) of an erased cell increases as P/E cycles increase [10]. After \( V_t \) crosses the reference voltage \( V_{READ} \), the cell cannot reliably operate as a memory element.
3. SSD architecture and flash memory structure

3.1 SSD architecture

Fig. 9 shows an SSD architecture [2], which consists of Processor, RAM, Host Interface, Buffer Manager, multiple Flash Controllers and Channels, and Flash-Chips. The Processor manages most of the SSD operations, such as request/response flow and logical-to-physical address mapping. The RAM consisting of SRAM and DRAM supports the functionalities of the Processor. The Host Interface provides a physical interface between the host and SSD, such as PCIe, SATA, SAS, etc. A Flash Controller handles commands from the Processor and manages the operations of Flash-Chips on its channel [2]. It is also responsible for transferring data between the Flash-Chips and the Buffer Manager. The multiple channels and flash-chips allow for exploitation of I/O parallelism [21] (see Section 9). The main responsibility of the Buffer Manager is to hold commands and data in order to handle requests and responses between the Host Interface and the Flash Controller.

Fig. 10 provides a more detailed illustration of a flash-chip in Fig. 9. Each flash-chip consists of dies, planes, blocks, and pages [16]. The I/O Control is used to communicate with the Flash Controller. The Control Logic issues
commands and corresponding addresses to the flash array. Data/Cache registers support read/write operations to NAND flash memory. Sections 3.1.1–3.1.3 explain the SSD read, write and advanced operations.

### 3.1.1 SSD read operation

An SSD read operation involves three phases: read set (RS), read from NAND (RN), and read data (RD). Fig. 11 illustrates the read operation. In the RS phase, a read command and the related address are sent to the Buffer Manager via the Host Interface. In the RN phase, the Buffer Manager first sends command/address to the Flash Controller then the Flash Controller transfers command to the corresponding flash-chip (i.e., setup). The flash-chip reads data from NAND flash array to the Data/Cache register, and the flash-chip is in busy state during this process. Afterward, the flash-chip transfers data to the Flash Controller, then the Flash Controller moves it to the Buffer Manager (i.e., data transfer). In the RD phase, the host system receives data from the Buffer Manager via the Host Interface using direct memory access (DMA) [11, 13].

### 3.1.2 SSD write operation

An SSD write operation consists of three phases: write set (WS), write data (WD), and write to NAND (WN). The SSD write operation is depicted in Fig. 12. In the WS phase, the host system sends a program command and the related address to the Buffer Manager via the Host Interface. Then, the

---

**Fig. 10** Micron 8 Gb flash-chip [16].
host system transfers data during the WD phase. During the WN phase, the Buffer Manager issues a program command/address, and then the Flash Controller sends command to the corresponding flash-chip (Setup). Afterward, data are transferred from the Buffer manager to the flash-chip via the Flash Controller (Data transfer). Then, the corresponding flash-chip writes data to its NAND flash array, and the state of this flash-chip is in busy state during the write operation. When the write operation is done, the flash-chip sends the status signal to notify the completion of the write operation [11, 13].

3.1.3 Advanced operations
A modern flash-chip provides advanced operations such as Two-plane Page Read, Two-plane Page Program, Multiple-die Page Read, Interleaved Program Page, Page Read Cache Mode, and Internal Data Move to exploit internal
parallelism within a flash-chip [16]. The Two-plane Page Read/Program operations allow for the two planes of the same die to be read/programmed at the same time. The Multiple-die Page Read and the Interleaved Program Page operations support the two dies of a flash-chip to be read and programmed, respectively, at the same time. The Page Read Cache Mode operation is used to speedup sequential read operations within a block. Finally, the Internal Data Move (copy-back) operation moves one page of data to another page within the same plane using the data register [16].

3.2 Structure of a flash device

There are two major flash memory structures: NAND flash and NOR flash. NOR flash has better read and write speeds than NAND flash. Although NOR flash memories provide higher performance, NAND flash memories are widely used in SSDs due to their small cell size (i.e., a low cost per bit). For this reason, NAND flash memory is mainly discussed in this paper.

A NAND flash-chip consists of pages, blocks, planes, and dies. A page consists of a number of flash cells that share the same word line, typically 512 B–16 KB. There is also additional spare area in each page to store a few bytes of management information such as ECC and mapping tables to support FTL operations [22]. The performance of read and program operations improves as the page size increases due to parallelism; however, it is limited by RC delay and lithography. In contrast to the read and program operations, erase operations are performed on a block granularity consisting of 32, 64, 128, or 256 pages. A large block size enhances array efficiency since more cells share the select gate and contact area. However, the number of valid data pages that need to be relocated during garbage collection increases, which is referred as write amplification (see Section 6) [23]. A plane is composed of 1024 or 2048 blocks. Finally, a die is made up of 2 or 4 planes, and there are 2 or 4 dies per flash-chip.

Fig. 10 shows an example NAND flash-chip, which is a Micron 8 Gb flash-chip with 2 dies. The capacity of an SSD is determined by the number of flash-chips, i.e., capacity = 8 Gb × n flash-chips. A flash-chip is composed of 2 dies, where each die contains 4 Gb flash memory. Each die is made up 2 planes, where a plane consist of 2048 blocks. Each block has 64 pages, and the page size is 2 KB [16]. Thus, the relationships for different granularity levels are summarized as follows:

1 page = 2048 + 64 (spare area) bytes
1 block = (2048 + 64) bytes × 64 pages = 132 KB
1 plane = 132 KB × 2048 blocks = 2112 Mb
1 die = 2112 Mb × 2 planes = 4224 Mb
1 flash-chip = 4224 Mb × 2 dies = 8448 Mb
1 SSD device = 8 Gb × n flash-chip

A NAND flash array is shown in Fig. 13 [13], which represents one of the planes in Fig. 10. A NAND string is a serial connection that shares the same bit line (i.e., a vertical line in Fig. 13, where all the transistors are connected through their source and drain), and a NAND page is a serial connection that shares the same word line (i.e., the horizontal line in Fig. 13, where all the transistors connected through their control gates). Since a NAND page shares the same control gate, the read and program operations are performed at a page granularity.

Programming individual cells within a page requires the program and program inhibit states shown in Fig. 14. For example, when data “0” needs to be programmed into the cell at word line (WL) 2 and bit line (BL) 1, i.e., (WL2, BL1), a high voltage is applied to WL2 and BL1 is grounded. Then, the control gate and substrate at the cell (WL2, BL1) has enough voltage difference (18 V) to cause FN tunneling. Meanwhile, the cell (WL2, BL2) is not a target for programming data “0”, but it shares the same control gate (i.e., WL2) with the cell (WL2, BL1). Thus, the program inhibit state is
required to prevent this cell from being programmed. This is done by applying a voltage (i.e., 8 V) to BL2 as shown in Fig. 14 such that it will not cause FN tunneling. Therefore, the cell (WL2, BL1) will be programmed with data “0” while the data at cell (WL2, BL2) will remain at “1”.

As mentioned before, NAND flash read operations are done at a page granularity. In SLC, $V_{READ}$ is applied to all the gates of a page being read, while the gates of the other pages are driven to $V_{PASS}$ (usually 5 V) to act as pass transistors. Then, the stored data can be determined based on whether or not the current passes through the transistor. Fig. 15 illustrates a read operation of a NAND flash array. Suppose the flash memory array has the following data: (WL1, BL1) = 1, (WL1, BL2) = 0, (WL1, BL3) = 1, (WL1, BL4) = 0, (WL2, BL1) = 1, (WL2, BL2) = 1, (WL2, BL3) = 0,
and \((WL2, BL4) = 0\). In order to read the target page on WL2 in Fig. 15, 
\(V_{\text{READ}}\) (2.5 V) is applied to WL2 and \(V_{\text{PASS}}\) (5 V) is applied to all other WLs. 
The cells \((WL2, BL1)\) and \((WL2, BL2)\) allow the current to pass since \(V_{\text{READ}}\) is higher than \(V_t\) of these cells (containing data “1”). On the other hand, the cells \((WL2, BL3)\) and \((WL2, BL4)\) obstruct the current because \(V_{\text{READ}}\) is lower than \(V_t\) of these cells (containing data “0”). Data in other pages (i.e., other WLs) are not affected when sensing the target WL because \(V_{\text{PASS}}\) is always higher than either data states as indicated in Fig. 15.

### 4. Flash translation layer

The flash translation layer (FTL) is an intermediate system made up of software and hardware that manages SSD operations as shown in Fig. 16 [1]. The FTL performs logical-to-physical address translation, garbage collection, wear-leveling, error correction code (ECC), and bad block management [4, 24]. In addition, FTL exploits data concurrency techniques, such as striping, interleaving, and pipelining, to achieve high throughput. Because of the need to perform garbage collection, wear leveling, and bad block management, the performance of SSDs drops off after all the flash blocks have been written to as shown in Fig. 17. This performance drop is called the write cliff [21]. As such, the FTL is crucial to SSD performance, and thus most of the system or architecture level research on SSDs is focused on FTL-related topics.

![Software architecture of SSD](image)
One of the main functions of FTL is to map logical addresses from the file system to physical addresses of the NAND flash memory. This address mapping will be discussed in Section 5. Afterward, the FTL needs to find new blocks to write data. In particular, update requests need to be written to free blocks due to the out-of-place update requirement and the pages of the original blocks are marked as invalid. As a result, many invalid pages can remain in the blocks. Garbage collection reclaims obsolete blocks that have more invalid pages than a given threshold, which will be explained in more detail in Section 6. Since flash memories have limited P/E cycles, wear-leveling is needed to evenly wear out flash devices in order to achieve longer lifetime. Wear-leveling techniques will be discussed in detail in Section 7. Despite the use of wear-leveling, flash memory blocks will eventually wear out. Therefore, these worn out blocks, referred to as bad blocks, need to be managed. Bad block management avoids the use of bad blocks due to the endurance problem and manufacturing faults. Bad block management techniques will be discussed in Section 8.

5. Address mapping

Address-mapping techniques are classified as page-level mapping, block-level mapping, and hybrid mapping based on the unit of flash memory granularity [8, 24]. Sections 5.1–5.3 discuss each of these address-mapping techniques.

Fig. 17 Write cliff [21].
5.1 Page-level mapping

Fig. 18 illustrates the page-level mapping scheme, where a logical address is mapped to a physical address at a page granularity. In this example, the file system sends the command `write(5, A)`, i.e., write data “A” to the logical page number (LPN) 5, and the page mapping table maps this to the physical page number (PPN) 10 in Block 2. The page-level mapping technique offers flexibility in choosing any valid pages to perform reads and writes. However, the page mapping table must maintain all the logical to physical mapping information for the entire flash memory, which requires a mapping table size of \([\lceil \log_2 P \rceil / 8] \times P \) bytes, where \( P \) is the number of pages in the SSD. For instance, a 1 TB SSD with 4 KB pages would require a mapping table with \( 2^{28} \) entries each with a 4-byte address. This leads to a mapping table size of 1 GB, which is too large to be stored in an SRAM [8, 9, 24]. Furthermore, the size of the mapping table increases as the size of flash memories increases.

5.2 Block-level mapping

In order to reduce the mapping table size, block-level mapping can be employed. Fig. 19 shows the block-level mapping scheme, where an LPN is converted to a physical block number (PBN) and an offset. In the example shown in Fig. 19, LPN 5 is divided by the number of pages per

![Fig. 18 Page-level mapping.](image_url)
block (in this case 4). The resulting quotient is referred to as the logical block number (LBN), i.e., LBN = 1, and the remainder represents an offset (i.e., 1) within the block for the corresponding page. Since the mapping table only contains logical-to-physical block mapping, the size of the mapping table is \( \frac{\log_2 B}{8} \times B \) bytes, where \( B \) represents the number of blocks in the SSD. For example, a 1TB SSD with 4KB pages and 64 pages per block requires 222 entries each with a 3-byte address. This leads to a mapping table size of 12 MB, which is a significant reduction compared to a table size of 1 GB for page-level mapping. However, a new (free) block has to be provided each time an update is made to the same logical page, and updating only part of a block causes expensive copy operations to move the valid pages from the old block to the new block. For example, update operations to pages 01, 05, 09, and 13 in Fig. 19 would each require a new block. Thus, the performance overhead due to “erase-before-write” is substantial because address mapping is performed at a block granularity [8, 24].

### 5.3 Hybrid mapping

In hybrid mapping, a logical block maps to a physical block using the block-level mapping scheme, and then the page-level mapping scheme is used to handle the updated pages within a block. Hybrid mapping maintains two types of blocks: data and log blocks [22]. Data blocks hold ordinary data and are managed using block-level mapping while log blocks are used for update requests and are managed using page-level mapping. For example,
if write(5, E0) is performed multiple times, the first write request is stored in the data block and then the subsequent update requests are written to the same log block. As another example, if the sequential write requests write(8, A0), write(9, B0), write(10, C0), and write(11, D0) are performed twice, the first set of sequential writes are written to the data block, while the second set of sequential writes are written to the same log block.

When all the log blocks are used up, merge operations have to be performed to secure new log blocks. Fig. 20 depicts three types of merge operations: switch merge, partial merge, and full merge. A switch merge is performed when update operations are performed sequentially to all the pages in a block [22, 25, 26]. This is illustrated in Fig. 20A, where update operations are performed on LPNs 0–3. After the update operations are completed, the original data block is erased (i.e., freed) and the log block becomes the new data

Fig. 20 Three types of merge operations [22, 25, 26]. (A) Switch merge; (B) partial merge; and (C) full merge.
block. The switch merge is the least costly operation because it only requires one erase operation. A partial merge shown in Fig. 20B is performed when a data block is partially updated, e.g., update operations are performed on LPNs 0–2. This requires additional copy operations to move the pages from the data block that were not updated to the corresponding entries in the log block. Then, the log block becomes the new data block and the old data block is erased. A full merge operation is performed when random writes occur, e.g., update operations are performed on LPNs 0, 0, 0, and 2 as shown in Fig. 20C. A full merge requires another new block to combine the valid pages from the original data block and the log block. The full merge is the most expensive among the three operations because it requires \(n\) copy operations (where \(n\) is equal to number of valid pages in a block) and two erase operations [25].

Hybrid mapping schemes take advantage of both page-level mapping and block-level mapping. There are several hybrid mapping schemes to manage log blocks, such as the Log-Block scheme [22], fully associativity sector translation (FAST) [24], the SuperBlock scheme [4], and locality aware sector translation (LAST) [25]. All these methods rely on log blocks; however, they differ on how the log blocks are handled. The following discusses these four techniques.

### 5.3.1 Log-Block

The seminal work on hybrid mapping is the Log-Block scheme, which maintains several log blocks in order to efficiently perform updates to the same logical pages [22]. This method is also referred to as the block-level associativity sector translation (BAST) scheme.

Fig. 21 illustrates the Log-Block scheme with four data blocks and three log blocks each with four pages. Suppose 12 consecutive write requests are made by the file system as shown. The first two requests \(\text{write}(5, E0)\) and \(\text{write}(6, F0)\) are mapped to Pages 01 and 02, respectively, in Block 00 using the block-level mapping scheme. However, the third and fourth requests \(\text{write}(5, E1)\) and \(\text{write}(5, E2)\) update Page 01 in Block 00, and thus a log block (Block 20) is allocated. Then, these two random write requests are written to Block 20, and the corresponding entry in the mapping table for the log block is updated. If LPN 5 is updated again later, these writes will be performed on Block 20 until it is fully filled up. In the same manner, the write requests \(\text{write}(8, A0)\), \(\text{write}(9, B0)\), \(\text{write}(10, C0)\), and \(\text{write}(11, D0)\) are written to Block 02. Afterward, the sequential write requests \(\text{write}(8, A1)\), \(\text{write}(9, B1)\), \(\text{write}(10, C1)\), and \(\text{write}(11, D1)\) update the Pages 08–11 in Block 02.
Thus, these requests are written to a new log block (Block 30) and the corresponding entry in the mapping table for the log block is updated.

In the Log-Block scheme, the cost of merge operations increases when frequent updates are made to the pages in different blocks [24]. For example, suppose write requests are issued to LPNs 1, 5, 9, 13, 1, 5, 9, and 13 as shown in Fig. 22. Due to one-to-one associativity between data and log blocks, the first sequence of write requests to LPNs 1, 5, 9, and 13 are written to different log blocks. This results in thrashing of log blocks, as shown in Fig. 22.

Fig. 21 The Log-Block scheme.

Fig. 22 Example of log block thrashing.
data blocks, and the second sequence of write requests for LPNs 1, 5, 9, and 13 will be written to different log blocks. However, all the log blocks will be used up by the time the second write request for LPN 13 is issued. Therefore, a merge operation has to be performed in order to secure a new log block. If write requests to LPNs 1, 5, 9, and 13 repeat, a merge operation is required for each write request leading to log block thrashing [24]. In addition, performance degradation occurs when one logical block is updated frequently with nonsequential write requests. For example, suppose write requests are made to LPNs 0, 1, 2, and 3, and then another write requests are made to LPNs 0, 1, 3, and 2 as shown in Fig. 23. Since the second sequence of write requests are not ordered, a full merge operation has to be performed to preserve the page ordering within a data block requiring four copy and one erase operations.

5.3.2 Fully associativity sector translation
The fully associativity sector translation (FAST) scheme utilizes many-to-one associativity to allow pages from multiple data blocks to be written to a single log block thereby circumventing the shortcoming of the Log-Block scheme [24]. FAST maintains two types of log blocks: one log block for sequential writes (SW log block) and a number of log blocks for random writes (RW log blocks). When an update request occurs, it is written to either the SW log block or a RW log block. An update request is written to the SW log block if either of
the following two conditions are satisfied: (1) the offset of the LPN being written to (i.e., LPN % 4) is 0 or (2) the write request is sequential with an offset that would fill the rest of the SW log block (i.e., 1, 2, or 3). Otherwise, it is written to a RW log block.

**Fig. 24** illustrates the FAST scheme with write requests to LPNs 1, 5, 9, 13, 1, 5, 9, and 13. The offsets of the second sequence of writes to LPNs 1, 5, 9, and 13 are not zeros, thus they are all updated in a RW log block (Block 30). Therefore, unlike the Log-Block scheme, FAST does not suffer from log block thrashing since a merge operation is not required until all the pages in a RW log block are fully utilized [24, 26]. Furthermore, when non-sequential write requests update all the pages of a block, FAST performs a switch merge operation instead of an expensive full merge operation required by the Log-Block scheme. This is illustrated in **Fig. 25**, which shows how FAST handles write requests to LPNs 0, 1, 2, 3, 0, 1, 3, and 2. FAST maintains the sequential order of offsets within the SW log block, and as a result only a switch merge is needed when the SW log block is reclaimed [24].

The FAST scheme efficiently utilizes log blocks by separating sequential and random writes. The authors in [24] showed that the performance of FAST with 4–8 log blocks is similar to the Log-Block scheme with more than 30 log blocks. Moreover, FAST reduces the garbage collection overhead by 30% compared to the Log-Block scheme [25].
5.3.3 SuperBlock

The SuperBlock scheme increases the chances of performing switch or partial merge operations instead of expensive full merge operations to minimize performance degradation. The SuperBlock scheme combines $N$ neighboring logical blocks into a superblock in order to exploit block-level spatial locality. The pages in a superblock can be freely mapped to any physical pages within the same superblock using a three-level page-mapping scheme [4]. The SuperBlock number and the page global directory (PGD) are stored in SRAM and used to map logical blocks. The page middle directory (PMD) points to the location of the page table (PT). Finally, the page table contains the physical block numbers and the physical page numbers. Both PMD and PT are stored in the spare area of the flash memory in order to reduce the overhead of SRAM. The following example shows how SuperBlock can increase the chance of performing switch or partial merge operations compared to the FAST scheme.

Fig. 26 shows switch and full merge operations in the SuperBlock scheme, where “hot” indicates pages that have been recently updated and are likely to be updated again in the near future, whereas “cold” indicates pages that have not been updated. When a switch merge operation is performed on a log block (Log Block 0) and a superblock consisting of two data
blocks (Data Blocks 0 and 1), the log block becomes the new data block which is then combined with the valid data block (i.e., Data Block 1) into a new superblock. This is shown in Fig. 26A. When a full merge operation is performed as shown in Fig. 26B, the log block and a free block are first grouped into a new superblock. Then, all the valid pages from the old superblock are copied to the free block in the new superblock.

The updated (i.e., hot) pages are grouped into the same block (i.e., Log Block 0) to increase the chances of performing switch merge operations. This is because the hot pages in a data block will likely be updated to a new log block in the near future. When all the pages in a data block are invalidated due to updates, the new log block can be converted to a data block without copy operations. If hot and cold pages are intermixed in the same block, a full merge operation is required to copy valid (i.e., cold) pages.

In order to illustrate the advantage of the SuperBlock scheme, Fig. 27 compares the merge operations for FAST and SuperBlock when LPNs 0–7 have already been written to and update requests are issued to LPNs 1, 3, 5, 6, 1, 3, 5, and 6. For FAST shown in Fig. 27A, the first sequence of write requests are written to a RW log block. After the second sequence of write requests are performed, all the RW log blocks are used up and a full merge operation has to be performed to reclaim RW log blocks. The full merge operation generates two new data blocks: One for LPNs 1 and 3 from RW Log Block 1 with LPNs 0 and 2 from Data Block 0 and another for...
LPNs 5 and 6 from RW Log Block 1 with LPNs 4 and 7 from Data Block 1. If the same set of write requests repeats, FAST has to perform a full merge operation each time.

On the other hand, the SuperBlock scheme combines the two data blocks into a superblock as shown in Fig. 27B. The first sequence of write requests are written to a log block (Log Block 0). Then, the second sequence of write requests requires a full merge operation. However, the hot pages (LPNs 1, 3, 5, and 6) in the Data Block 0 and the cold pages (LPNs 0, 2, 4, and 7) in the Data Block 1 are separated by dynamically rearranging the pages as the new SuperBlock 0 shown in Fig. 27B. If the same sequence of write requests repeats, a switch merge instead of a full merge can be performed. Therefore, the SuperBlock scheme reduces the garbage collection overhead by reducing the frequency of full merge operations.

The authors in [4] showed that the garbage collection overhead is reduced by 23% when the superblock size $N$ is 32 compared to when there is no superblock (i.e., $N$ is one). They also show that the SuperBlock scheme (with $N$ equal to 32) decreases the garbage collection overhead by 35% compared to FAST [4]. However, the read performance of the SuperBlock scheme can be reduced due to the three-level page mapping scheme because PMD and PT need to be read from the spare area of the flash memory to generate the mapping table.

Fig. 27 Comparison of merge operations for FAST (A) and SuperBlock (B).
5.3.4 Locality-aware sector translation

The locality-aware sector translation (LAST) scheme was proposed in order to consider both spatial and temporal locality of access patterns [25]. This scheme exploits spatial locality by using SW log blocks (similar to FAST) and temporal locality by distinguishing blocks as hot and cold (similar to SuperBlock).

Fig. 28 shows the architecture of the LAST scheme. The Locality Detector distinguishes between random and sequential write requests. If the length of sequential write requests is longer than a threshold (i.e., 8), they are written to SW log blocks; otherwise, they are written to RW log blocks. Therefore, multiple SW log blocks are used to log long sequential writes to exploit spatial locality, while RW log blocks are used to exploit temporal locality by distinguishing hot and cold regions to avoid expensive full merge operations.

For random writes, data are distinguished into hot and cold partitions based on how often they are updated. The hotness of each page is determined based on its update interval, which is determined by the distance between updates to the same logical page. For example, if writes are performed on LPNs 0, 3, 2, and 0, the update interval of LPN 0 is 2 because there are two other requests (LPNs 3 and 2) between the first and second writes to LPN 0. If the update interval of a page is less than $k$ as a criterion, it is considered as a hot page; otherwise, it is treated as a cold page. Fig. 29

![Fig. 28 The architecture of LAST [25].](image-url)
compares a scheme with no partitioning with a scheme that partitions pages into hot and cold. When partitioning is not used, the requests are sequentially written to the random log blocks regardless of whether pages are hot or cold. On the other hand, when hot and cold pages are partitioned, the second write to LPN 0 is performed on the hot partition because its update interval is 2, which is less than $k$ (i.e., $k = 4$). Similarly, writes to LPNs 1 and 8 are also considered as hot [25].

LAST exploits both spatial and temporal locality by using SW log blocks and hot/cold partitions. However, it cannot efficiently detect a workload consisting of a series of short sequential writes instead a series of long sequential writes [26]. For example, if the length of sequential writes is shorter than the threshold, they are not stored in SW log blocks. Furthermore, the threshold should be carefully selected. If the threshold is too long, many sequential writes will be treated as random writes. If the threshold is too short, many random writes will be handled using the SW log blocks. The authors in [25] show that for certain applications LAST reduces garbage collection overhead by 86%, 81%, and 46% compared with the Log-Block scheme, FAST, and SuperBlock, respectively.

5.3.5 Demand-based flash translation layer

The Demand-based flash translation layer (DFTL) scheme was proposed to reduce the mapping table size in the SRAM for the page-level mapping scheme [26]. The basic idea is to store the page-based mapping table in the flash memory and then cache the entries on demand in the cached mapping.
table (CMT) in the SRAM. *Translation blocks* in the flash memory hold the entire page-level mapping table, and the *global translation directory* (GTD) in the SRAM tracks these translation blocks. Storing the mapping table in the flash memory causes performance overhead for read requests because the translation blocks need to be accessed first from the flash memory to bring the CMT entries into the SRAM. Therefore, the DFTL stores small GTD and CMT in the SRAM instead of a large page-based mapping table. The DFTL improves response time by 78% compared to FAST [26]. However, storing the mapping table in the translation blocks causes a serious performance overhead for random read requests even with CMT. Furthermore, extra write operations are required to update the translation blocks.

6. Garbage collection

Erase operations are performed at a block granularity due to their long latency and the limitations of flash devices. When write requests are issued for logical pages, they have to be written to free pages (i.e., out-of-place updating), and the original pages are marked as invalid. As a result, many invalid pages are generated and these obsolete pages cannot be overwritten due to the characteristics of flash memories as discussed in *Section 2*. Therefore, the FTL has to prepare and maintain a pool of free blocks for write operations. *Garbage collection* (GC) is the reclamation process that invalidates obsolete blocks and make them available for future write operations.

The main issues for garbage collection are minimizing its cost and maximizing the reclaimed space, and the important metrics used to evaluate different garbage collection schemes are the write amplification factor and the garbage collection efficiency. The *write amplification factor* is the ratio of the actual number of written pages and the number of page writes [23]. On the other hand, the *garbage collection efficiency* is defined as the average number of invalid pages in each victim block to be erased [27]. An optimized garbage collection algorithm reduces write amplification, thus improving the performance and lifetime of flash devices. For example, *Fig. 30* illustrates the importance of *victim selection* in garbage collection [28]. In Scenario 1, Blocks 1 and 2 are chosen as victim blocks, and the valid pages in these two blocks are moved to Block 7. Afterward, Blocks 1 and 2 are erased to make them free blocks. On the other hand, in Scenario 2, Blocks 1 and 6 are chosen as victim blocks, which do not have any valid pages, and thus...
no data needs to be moved. As can be seen, even though both Scenario 1 and 2 have the same data, Scenario 2 leads to a more efficient garbage collection process than Scenario 1 due to better victim selection. Existing garbage collection algorithms differ on how victim section is performed, which include first in first out (FIFO), Greedy, Windowed, and $d$-choice [29]. Fig. 31 illustrates these algorithms, and Sections 6.1–6.4 discuss each of these techniques.

### 6.1 FIFO GC algorithm

The *FIFO GC algorithm* chooses blocks to be reclaimed in a cyclic manner where the first written block is erased first [29]. For example, in Fig. 31, the FIFO GC algorithm will reclaim Blocks 1 through 6 sequentially, even though over 50% of the pages in Blocks 1 and 2 are still valid. Therefore, the FIFO GC algorithm is easy to implement, but it wastes many valid pages and increases write amplification [29].
6.2 Greedy GC algorithm

In contrast to the FIFO GC algorithm, the Greedy GC algorithm selects blocks that contain the fewest number of valid pages [30]. As shown in Fig. 31, the Greedy GC algorithm will select Blocks 3, 4, 6, 5, 2, and 1 in order representing the fewest number of valid pages, which allows Blocks 1 and 2 to be used longer until they are reclaimed. Hence, the Greedy GC algorithm results in the least amount of write amplification during random write operations [30]. However, the implementation overhead of the Greedy algorithm is significantly higher than the FIFO algorithm.

6.3 Windowed GC algorithm

The Windowed GC algorithm take advantages of both FIFO and Greedy algorithms [29]. The algorithm selects a window containing a list of blocks using the FIFO algorithm. From this selected window, the algorithm then chooses the block that has the fewest number of valid pages using the Greedy algorithm. Fig. 31 illustrates the Windowed GC algorithm with a window size of two blocks. In this example, Blocks 4 and 5 are first chosen using the FIFO algorithm. Then, Blocks 4 and 5 are selected sequentially using the Greedy algorithm. The most appropriate window size is a design trade-off between simplicity of implementation and resulting write amplification. As the window size increases, the complexity increases. On the other hand, as the window size decreases, the amount of write amplification increases [29].
6.4 d-Choice GC algorithm

The d-choice GC algorithm is similar to the Windowed GC algorithm, but the set of d blocks is chosen randomly instead of in a FIFO manner [29]. From this chosen d blocks, the algorithm reorders the blocks for garbage collection using the Greedy algorithm. For example, in Fig. 31, Blocks 1, 2, and 3 are selected randomly, and then they are reordered based on the fewest invalid pages. As a result, the d-choice GC algorithm selects Blocks 3, 2, and 1 in sequence [29].

7. Wear leveling

As discussed in Section 2, flash devices have a finite number of P/E cycles as a reliable memory. If a subset of blocks are used excessively, then these blocks would wear out sooner and, as a result, the total storage capacity will decrease over time. This decrease in capacity seriously impacts the performance because if there are not enough free blocks to support write requests, garbage collection needs to be performed more frequently to secure additional free blocks degrading performance [27]. Therefore, uneven wearing of flash blocks leads to not only capacity degradation but also performance degradation. To combat this problem, wear-leveling algorithms try to wear out flash blocks more evenly. Dynamic wear leveling techniques select the blocks with the lowest number of P/E cycles when new write requests are issued [1, 31]. However, dynamic wear leveling operates only on the currently active blocks, and does not consider the blocks that have not been updated for a long time, such as the ones that contain the operating system. Thus, static wear leveling techniques changes the state of inactive blocks to active blocks [1].

Fig. 32 illustrates the reason why both static and dynamic wear leveling are required to avoid worn out flash blocks. The blocks that are hot are frequently updated, while the blocks that are cold are rarely updated. In this example, the P/E cycle counts for the cold block and the hot block are 1 k and 5 k, respectively. If the dynamic wear leveling technique is used exclusively for 5 k P/E cycles, only the hot block will be worn out because the cold block has not been updated as shown on the left side of Fig. 32. In contrast, if both static and dynamic wear leveling are used, the P/E cycle limit of 10 k (in case of SLC) can be avoided by swapping the hot block with the cold block and distributing the number of P/E cycles between the two blocks as shown on the right side of Fig. 32.
Numerous wear leveling techniques have been proposed in the literature [5, 31–33]. Sections 7.1 and 7.2 discuss the Evenness-Aware and the dual-pool algorithms. The concept behind these two algorithms are similar, but how they are triggered is different.

### 7.1 Evenness-aware algorithm

The *Evenness-Aware algorithm* was proposed to improve the endurance using the block erasing table (BET) that maintains information about the number of P/E cycles within a certain period of time [5]. The algorithm minimizes the difference in P/E cycles between any two blocks using a cyclic-queue-based scanning procedure to evenly distribute P/E cycles among the blocks. A software leveler is triggered whenever a block is recycled by the garbage collection process. The algorithm then performs static wear-leveling based on a preset condition, which is the difference between the maximum and the minimum number of P/E cycles. If this difference is larger than a preset condition, garbage collection with static wear leveling is performed and information in the BET is updated. Therefore, Evenness-Aware algorithm avoids static data remaining at any block for a long time [5].

### 7.2 Dual-pool algorithm

The *dual-pool algorithm* maintains two types of data pools: cold pool and hot pool [31]. The *cold pool* contains blocks that are updated infrequently, such as memory blocks that store the operating system, while the *hot pool* contains...
blocks that are frequently updated. The P/E cycle counts increase slowly for the blocks in the cold pool, while the P/E cycle counts increase quickly for the blocks in the hot pool due to frequent updating. The main idea of the dual-pool algorithm is to monitor the P/E cycle count of each block, and then swap a block from the hot pool with a block from the cold pool when the P/E cycles of the hot block is larger than that of the cold block by a predefined threshold [31].

8. Bad block management

Flash blocks can become bad due to faults during production process or become worn out after a limited number P/E cycles. Bad block management avoids these bad blocks to maintain high reliability [6, 7]. Sections 8.1 and 8.2 discuss the process of bad block management, which consists of bad block recognition and bad block replacement.

8.1 Bad block recognition

There are a couple of ways to detect bad blocks: Using the code “FFh” and error correction code (ECC) [7]. “FFh” is a special code included with each valid block (located at either 1st or 2nd page of a valid block). Note that “FFh” basically represents a byte that has been erased (i.e., all the bits represent data “1”). Therefore, if this “FFh” is missing, it means that an erase failure occurred.

8.2 Bad block replacement

The FTL creates a table of bad blocks from the bad block recognition. A bad block replacement is required when the target address of a request matches with the address of a bad block. There are two types of bad block replacement schemes: Skip Block and Reserve Block methods. The Skip Block method avoids the use of a bad block by writing the data to the block right after the bad block [7]. The Skip Block method is depicted in Fig. 33. In this example, the 2nd block in the device is bad; therefore, this block is skipped and the 3rd block is chosen instead. In the Reserve Block method, a good block from the reserved block area (RBA) is chosen to write the data when a bad block is recognized [7]. The Reserve Block method is illustrated in Fig. 34. In this example, the 2nd block is detected to be a bad block as in the previous example, thus the data 2 is written to a good block (referred to as a reservoir block) from the reserved block area.
9. **SSD parallelism techniques**

An SSD exploits parallelism within the flash memory to increase the overall bandwidth and amortizes the cost of long latency operations. This is achieved by employing techniques such as channel striping, flash-chip pipelining, die interleaving, and plane sharing in order to achieve high performance [21, 34, 35]. Sections 9.1–9.5 discuss these methods in detail.

---

Fig. 33  Skip Block method.

Fig. 34  Reserved block method.
9.1 Channel striping

The Channel Striping technique exploits channel-level parallelism in a multi-channel architecture. This is illustrated in Fig. 35, where multiple-requests are distributed over the multiple channels in a round-robin fashion. For example, the four write requests are allocated to Channels 0–3 as shown in Fig. 35 [2, 3, 21, 35].

9.2 Flash-chip pipelining

The Flash-chip pipelining technique exploits flash-chip-level parallelism by utilizing multiple flash-chips on a channel as shown in Fig. 36. In this example, the four write requests are assigned to the four flash-chips on Channel 1 [35].

9.3 Die interleaving

The die interleaving technique exploits die-level parallelism using the interleaved program page command. This is depicted in Fig. 37, where Request 0 and Request 1 are allocated to Die 0 and Die 1, respectively, on the same flash-chip (i.e., Flash-chip 0) [16, 35].
Requests ⇒ Multiple Flash-chips

Fig. 36 SSD flash-chip pipelining.

Requests ⇒ Multiple dies

Fig. 37 SSD die interleaving.
9.4 Plane sharing

The plane sharing technique exploits plane-level parallelism using the Two-plane Page Program command. This is shown in Fig. 38, where Request 0 and Request 1 are assigned to Plane 0 and Plane 1, respectively, on Die 0 in Flash-chip 0 [16, 35].

9.5 Combining parallelism techniques

All three techniques (channel striping, flash-chip pipelining, and die interleaving (or plane sharing)) can be combined together to exploit multiple levels of parallelism. This is illustrated in Fig. 39, where there are 8 requests. The Requests 0–3 and 4–7 are assigned to different channels according to the channel striping technique. The requests 0–1 (or 4–5) and 2–3 (or 6–7) are allocated to different flash-chips using the flash-chip pipelining technique. Finally, Requests 0 and 1 (or 2 and 3, 4 and 5, 6 and 7) are assigned to different dies using the die interleaving technique. Therefore, combining these parallelism techniques can provide significant improvement in throughput.
and performance [21]. In addition, the order of how these parallelism techniques are applied is important for the performance of SSDs. This will be discussed in the following section.

10. Page allocation

As mentioned in Section 9, write requests can be distributed among multiple channels, chips, dies, and planes using channel striping, flash-chip pipelining, die interleaving, and plane sharing, respectively. The distribution of write requests requires the allocation of pages, and thus the order of page allocation among channels, flash-chips, dies, and planes is important for improving the performance of not only writes but also subsequent reads and updates [35, 36].

Static allocation assigns logical pages to free physical pages in the flash memory based on a predetermined order of channel, flash-chip, die, and plane. On the other hand, dynamic allocation considers P/E cycles of blocks, idle/busy state of flash-chips, and/or idle/busy state of channels. Studies have shown that static allocation schemes consistently outperform dynamic allocation schemes for read operations. In contrast, dynamic allocation schemes are designed to improve the lifetime of SSDs [36].
The performance of static allocation schemes can be evaluated based on a predetermined priority [35]. There are 24 possible static allocation schemes because the permutation among channel, flash-chip, die, and plane is 4! The channel-first allocation exploits channel-level parallelism using channel striping, while the flash-first allocation maximizes the benefit of flash-chip-level parallelism using flash-chip pipelining. The die-first allocation takes advantage of die-level parallelism using die interleaving. The plane-first allocation scheme exploits plane-level parallelism using plane sharing, where flash operations are activated concurrently on multiple planes [35].

The best allocation schemes for channel-first, flash-first, die-first, and plane-first are shown in Fig. 40A–D, respectively. The arrows indicate the order of how the resources are allocated. For example, “Channel → Flash-chip → Die → Plane” (CFDP) indicates that the request is stripped in order across multiple channels, then across multiple flash-chips, then across multiple dies, and finally across multiple planes as shown in Fig. 40A. The channel-first and the flash-first allocation policies are better for latency.
sensitive applications, while the die-first and the plane-first allocation policies are better for throughput sensitive applications. Among these page allocation strategies, the optimal page allocation strategy is the Plane $\rightarrow$ Flash-chip $\rightarrow$ Channel $\rightarrow$ Die (PFCD) allocation scheme [35].

11. Recent research trend

Performance, lifetime, and reliability are all important factors for SSDs, and tradeoffs exist among these factors. This section focuses on recent research efforts on improving these factors. Table 1 provides a taxonomy of various research efforts on FTL functionalities as well as non-FTL-related approaches to improve performance, lifetime, and reliability. Since lifetime and reliability of SSDs are highly related, they are combined into the same category.

11.1 Performance

Performance is the most important metric for SSDs in enterprise applications. Therefore, most FTL schemes focus on boosting performance of SSDs. Currently, garbage collection is the main culprit for performance degradation. Sections 11.1.1–11.1.10 present recent techniques that focus on performance.

11.1.1 Host Interface I/O Scheduler

The Host Interface I/O Scheduler (HIOS) is a garbage collection and quality of service (QoS) aware FTL [21]. This is achieved by redistributing the garbage collection overhead across noncritical I/O requests as well as performing Channel QoS-aware scheduling. The authors in [21] observed that the worst-case latency of SSDs due to garbage collection and channel contention can vary between 210 and 4670 ms.

As shown in Fig. 41, HIOS classifies I/O activities based on their current level of deadline satisfaction. If an I/O request takes longer than the require deadline, it is defined as a negative activity (i.e., critical I/O request). If an I/O request takes less than but close to the required deadline, it is categorized as a neutral activity. If an I/O request takes much less than the required deadline, it is classified as a positive activity (i.e., noncritical I/O request). The main idea of HIOS is to predict which I/O requests are negative activities and distributes their overhead across positive activities.

In order to predict whether an I/O request is a negative activity, HIOS analyzes the length of the I/O request (i.e., the number of writes) and its
<table>
<thead>
<tr>
<th>FTL operations</th>
<th>Performance</th>
<th>Lifetime and reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address mapping and allocation</td>
<td>- Real-time flash translation layer (RFTL) [37]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Multilevel associated sector translation (MAST) [38]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Zone-based flash translation layer (Z-MAP) [39]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Convertible flash translation layer (CFTL) [40]</td>
<td></td>
</tr>
<tr>
<td>Garbage collection</td>
<td>- Host Interface I/O Scheduler (HIOS) [21]</td>
<td>- Stable Greedy garbage</td>
</tr>
<tr>
<td></td>
<td>- Real-time flash translation layer (RFTL) [37]</td>
<td>collection [41]</td>
</tr>
<tr>
<td></td>
<td>- Lazy real time garbage collection (Lazy-RTGC) [43]</td>
<td>- Garbage collection aware</td>
</tr>
<tr>
<td></td>
<td>- Stable Greedy garbage collection [41]</td>
<td>striping (GCAS) [42]</td>
</tr>
<tr>
<td></td>
<td>- Garbage collection aware striping (GCAS) [42]</td>
<td></td>
</tr>
<tr>
<td>Wear leveling</td>
<td>- BER-based wear leveling and bad block management [33]</td>
<td>- Zombie NAND [17]</td>
</tr>
<tr>
<td>Bad block management</td>
<td>- Zombie NAND [17]</td>
<td></td>
</tr>
<tr>
<td>Others</td>
<td>- Willow [44]</td>
<td>- Self-healing SSDs [45]</td>
</tr>
<tr>
<td></td>
<td>- An Effective Page Padding method for RAM buffer algorithms [46]</td>
<td>- Dynamic program and erase</td>
</tr>
<tr>
<td></td>
<td>- Harey Tortoise [48]</td>
<td>scaling (DPES) [47]</td>
</tr>
<tr>
<td></td>
<td>- Flash correct-and-refresh (FCR) [49]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Retention optimized reading (ROR) and retention failure recovery (RFR) [50]</td>
<td></td>
</tr>
</tbody>
</table>
LPNs, then determines the target physical block(s) to be written to. Moreover, if the number of free pages available in the target physical block(s) is less than the length of the I/O request, additional blocks need to be freed in order to satisfy the request. Therefore, the garbage collection time is also estimated. Based on this, HIOS predicts whether or not this I/O request is a negative activity. If it is predicted as a negative activity, its garbage collection overhead is scheduled across the available time slots of positive activities.

Channel QoS-aware scheduling in HIOS reorders I/O requests based on Empty Channel First. This is illustrated in Fig. 42, which compares FIFO I/O scheduling and Channel QoS-aware scheduling. Suppose Requests 0–5, which are neutral activities, are issued to Channels 0, 1, 0, 2, 0, and 3, respectively. Fig. 42A illustrates the FIFO I/O scheduling, where Request 2 has to wait until Request 0 is completed due to channel contention, and this delay affects Requests 3–5. On the other hand, the Channel QoS-aware scheduling shown in Fig. 42B removes channel contention by interchanging Requests 2 and 3 because Channel 2 is empty. HIOS has shown to reduce worst-case latencies by 86.6% compared to FIFO I/O scheduling used in commercial systems [51].

11.1.2 Real-time flash translation layer

The real-time flash translation layer (RFTL) scheme considers real-time service guarantees by focusing on reducing worst-case response time [37]. This is achieved by employing a distributed partial garbage collection scheme with the real-time task scheduler and a real-time aware hybrid addressing mapping technique.
As the name suggests, the distributed partial garbage collection scheme divides the garbage collection process using a time-cost model that considers the latencies of read, write, and erase operations as well as the number of valid pages that need to be copied from the victim block(s) to a new block. For

Fig. 42 Comparison of FIFO I/O scheduling (A) and channel QoS-aware scheduling (B).

As the name suggests, the distributed partial garbage collection scheme divides the garbage collection process using a time-cost model that considers the latencies of read, write, and erase operations as well as the number of valid pages that need to be copied from the victim block(s) to a new block. For
example, Fig. 43 illustrates the garbage collection process, which can be divided (i.e., distributed) into copy and erase operations. Suppose the latencies of read, write, and erase operations are 30 μs, 300 μs, and 3 ms, respectively, and the victim block has 18 valid pages that have to copied to a free block before the erase operation resulting in a total garbage collection time of 8940 μs (i.e., 5940 μs for read and write operations and 3000 μs for the erase operation). This process can be divided into copy operation 1 (9 read and 9 write operations), copy operation 2 (9 read and 9 write operations), and one erase operation requiring 2970 μs, 2970 μs, and 3000 μs, respectively. Afterward, the real-time task scheduler interleaves the distributed partial garbage collection workload with other pending tasks (i.e., write operations w1, w2, w3, and w4) to reduce the worst-case response time to 3300 μs as shown in Fig. 43.

Interleaving write requests with a distributed partial garbage collection can cause a coherence problem. This situation is explained using the example shown in Fig. 43. Suppose the write request w2 is to a valid page in the victim block, which has already been copied to a new block during copy operation 1. This causes a synchronization problem where the page that has been copied is no longer valid and the write request w2 must be properly reflected.

In order to handle this situation, RFTL utilizes the real-time aware hybrid address mapping, which maintains three types of physical block—primary, replacement, and buffer blocks as shown in Fig. 44. Initial write requests
are first served by a primary block, which is similar to a data block in existing hybrid mapping schemes. The update requests are performed to a replacement block, which is similar to a log block in existing hybrid mapping schemes. When a garbage collection is triggered for the primary block, all of its valid pages are migrated to the corresponding replacement block during copy operations 1 and 2 shown in Fig. 43. Meanwhile, the pending write requests to this logical block (i.e., w2 and w3 in Fig. 43) are allocated to the corresponding buffer block. Afterward, the replacement block becomes the new primary block, and the freed primary block becomes the new replacement block. When the buffer block becomes full, then it becomes the new replacement block and the freed primary block becomes the new buffer block. These three blocks periodically change their roles in order to provide a write space during the distributed garbage collection process. A page-level mapping table is required to freely update pages among the three types of blocks. The page-level mapping table indicates which physical page belongs to one of the three types of physical blocks, and is stored in the spare area to reduce overhead in SRAM.

The RFTL scheme reduces worst-case latency of garbage collection. However, RFTL requires extra space to manage the three types of blocks. Furthermore, storing page-level mapping table in the spare area may effect latency of read operations.

Fig. 44 The hybrid address mapping table for RFTL [37].
11.1.3 Lazy real time garbage collection
The lazy real time garbage collection (Lazy-RTGC) scheme defers garbage collection as long as possible to optimize both the average and the worst-case system response time. This is achieved using on-demand page-level address mapping and partial garbage collection [43]. On-demand page-level address mapping stores the entire page-level address mapping table in the translation blocks and then the table entries are selectively cached in SRAM to provide the advantage of page-level mapping with a minimum SRAM cost [26]. On the other hand, partial garbage collection is similar to the distributed partial garbage collection scheme in RFTL (see Section 11.1.2), but the address mapping scheme and the scheduling method are different.

Unlike RFTL that uses hybrid mapping, Lazy-RTGC can handle update requests during a partial garbage collection because page-level mapping scheme is used. Moreover, it does not need extra space compared to RFTL. For these reasons, the authors in [43] show that the number of erase operations is reduced by 67.4% compared to RFTL and the SRAM overhead is reduced by 90.0% compared to pure page-level mapping.

However, read operations of Lazy-RTGC incur longer latency because the mapping table is stored in the translation blocks need to be read from flash memory. Furthermore, the mapping table in the translation blocks needs to be updated with every write request, which causes frequent garbage collection operations.

11.1.4 Stable Greedy GC
The Greedy GC scheme discussed in Section 6 is highly complex and scales poorly. Furthermore, it does not consider whether or not a block is hot during the victim block selection. In contrast, the Stable Greedy GC scheme takes into account hot and cold blocks during the victim block selection process [41]. This is achieved by considering page lifetime, which is the time between when new data is written to a page and when the page is invalidated due to an update. A page is considered to be hot if its lifetime is short, which is similar to the update interval used in LAST (see Section 5.3.4) [25]. Another important factor used in the Stable Greedy GC scheme is Block Dormant Period, which is the interval between the current time and when a page in a block was last invalidated. A short Block Dormant Period for a block indicates that the pages in this block are likely to be updated in the near future. The Stable Greedy GC scheme avoids selecting a victim block with a short Block Dormant Period because it is considered as an unstable block. Therefore, deferring the choice of these unstable blocks as victim blocks will reduce the number of copy operations during garbage collection.
Fig. 45 compares victim block selection for Greedy and Stable Greedy, where Lv indicates the level of invalid pages, i.e., the blocks in Lv 1 have the most number of invalid pages followed by Lv 2 and so on. The Greedy scheme first chooses block B0 as the victim block because it has the most number of invalidated pages. Then, blocks B1, B2, and B3 are chosen as candidates due to the number of invalid pages these blocks have, and so on. In contrast, the Stable Greedy scheme avoids blocks B0, B2, B3, and B6 (highlighted by green regions) because their Block Dormant Periods are less than average. Therefore, blocks B1 and B4 are chosen as candidates, and then block B1 is selected as the victim block because it has more invalid pages than block B4. Therefore, Stable Greedy is an efficient and adaptive garbage collection scheme that has shown to outperform previous garbage collection algorithms under various workloads [41].

11.1.5 Multilevel associated sector translation

Previous hybrid mapping techniques such as the Log-Block scheme, FAST, LAST, and SuperBlock discussed in Section 5 do not consider workload characteristics. In contrast, multilevel associated sector translation (MAST) is a workload-aware address mapping scheme that dynamically detects hot and cold pages in order to avoid unnecessary merge operations [38].
Fig. 46 shows the architecture of MAST, which has three types of log areas—Locality Detection Area, Hot Area, and Cold Area. The Locality Detection Area has a small number of log blocks consisting of Locality Blocks and Sequential Log Blocks. The Locality Detection Area is used to distinguish between hot and cold pages for write requests and is similar to LAST (see Section 5.3.4), and Locality Blocks and Sequential Log Blocks follow the basic functionalities of RW and SW log blocks in FAST, respectively [24].

Initial sequential write requests in MAST are allocated to the Sequential Log Blocks. On the other hand, initial random write requests are allocated to the Locality Blocks in order to distinguish between hot and cold pages. When update requests are issued to pages in the Locality Blocks, they are considered as hot pages. Then, these update requests are written to the Hot Area.

The Cold Area receives cold pages from Locality Blocks and the Hot Area. When all the blocks in the Locality Detection Area are used up, all the valid pages are considered as cold pages and, as a result, they are copied to the Cold Area. If there is no more space in the Hot Area, its valid pages are no longer considered as hot and are moved to the Cold Area.
When Sequential Log Blocks in the Locality Detection Area and log blocks in the Cold Area are all used up, a merge operation is performed. Therefore, MAST reduces unnecessary copy operations by storing stable data such as sequential and cold data in data blocks, while unstable data such as hot data are stored in log blocks. As a result, it provides better performance in PC application workloads compared to FAST and LAST by 7% and 3%, respectively.

11.1.6 Dynamic hybrid address mapping techniques

Hybrid mapping techniques discussed in Section 5.3, such as the Log-Block scheme, FAST, LAST, SuperBlock, and MAST, are considered to be static schemes. Recently, dynamic hybrid mapping techniques that switch between page-level and block-level mappings have been proposed. These include zone-based flash translation layer (Z-MAP) \[39\] and convertible flash translation layer (CFTL) \[40\].

The Z-MAP scheme utilizes two types of zones: Page-mapping Zone and Block-mapping Zone. This is shown in Fig. 47. The Page-mapping Zone is used to store random data to better manage a large number of partial updates. On the other hand, the Block-mapping Zone is used store sequential data to reduce the size of the mapping table. These zones are dynamically allocated based on the workload. The Z-map scheme avoids expensive merge operations by classifying the workload before it is stored into flash.
memory, thereby improving the garbage collection efficiency. Therefore, Z-MAP is a workload adaptive technique that not only improves performance by reducing the garbage collection overhead but also reduces the mapping table size by using the Block-mapping Zone [39]. The authors in [39] show that Z-Map improved performance by 76%, the mapping table size is reduced by 81%, and the garbage collection overhead is reduced by 88% compared to FAST [24] and DFTL [26].

The CFTL scheme also dynamically switches between block-level mapping and page-level mapping based on data access patterns, i.e., block-level mapping is selected for read intensive data, while page-level mapping is adopted for write intensive data. In order to reduce the size of the page-level mapping table for write intensive workloads, CFTL employs a spatial locality-aware caching mechanism and adaptive cache partitioning [40]. The spatial locality-aware caching mechanism stores the mapping information only for the first page of a block of consecutive pages. For example, suppose LPNs 0, 1, 2, and 3 are mapped to PPNs 110, 111, 112, and 113, respectively. Then, the mapping information is maintained only for LPN 0 and PPN 110 and the number of consecutive pages (i.e., 4) as shown in Fig. 48. The adaptive cache partitioning is used to adjust the size of page-level and block-level mapping tables according to the workload. If the access pattern contains mainly write requests, the size of the page-level mapping table is increased. On the other hand, when the access pattern contains mainly read requests, the size of the page-level mapping table is decreased. As a result, CFTL outperforms DFTL [26] by up to 24% and 4% for read intensive workloads and write intensive workloads, respectively [40].

11.1.7 Write buffer management schemes
As discussed in Section 5, the mapping table is stored in the SRAM to minimize the address translation overhead. There are also techniques that use the

Page-level mapping table

<table>
<thead>
<tr>
<th>LPN</th>
<th>PPN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>110</td>
</tr>
<tr>
<td>1</td>
<td>111</td>
</tr>
<tr>
<td>2</td>
<td>112</td>
</tr>
<tr>
<td>3</td>
<td>113</td>
</tr>
</tbody>
</table>

Cached page mapping table

<table>
<thead>
<tr>
<th>LPN</th>
<th>PPN</th>
<th>Consecutive</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>110</td>
<td>4</td>
</tr>
</tbody>
</table>

Spatial locality-aware caching

![Fig. 48 The spatial locality-aware caching for CFTL.](image-url)
Write requests
01. Write(0, A0) / 02. Write(0, A1) / 03. Write(0, A2) / 04. Write(3, D0)

SRAM as a write buffer to increase the performance of random writes [52]. In order to illustrate the importance of a write buffer, suppose update requests are issued to LPNs 0 (A0), 0 (A1), 0 (A2), and 3 (D0) after LPNs 0–3 have already been written to Data Block 0 as shown in Fig. 49. The Log-Block
scheme (see Section 5.3.1) without a write buffer updates LPNs 0, 0, 0, and 3 to Log Block 0, and then a full merge operation is performed as shown in Fig. 49A. Therefore, the Log-Block scheme without a write buffer requires 4 copy and 2 erase operations. On the other hand, the Log-Block scheme with a write buffer shown in Fig. 49B updates LPNs 0, 0, 0, and 3 to the write buffer in the SRAM instead of the flash memory. The write buffer holds the updated data until it is evicted and written back to the flash memory, which reduces write traffic and delay. Therefore, properly selecting a victim block is crucial for performance of SSDs and a number of schemes have been proposed [53–55].

The problem with only focusing on the victim block selection is that a nonsequential LPN order may cause similar number of copy and erase operations as in the case without a write buffer. For example, in Fig. 49B, only LPNs 0 and 3 need to be written to the Log Block 0 during eviction. However, a full merge operation will be required afterward due to nonsequential LPN order of the Log Block 0. For this reason, a number of methods have been proposed to reduce the number of full merge operations and/or erase operations.

The Block Padding Least Recently Used (BPLRU) scheme utilizes the page padding technique to reduce the number of full merge operations [52]. This technique pads the log block in the write buffer with valid pages from the data block in the flash memory as shown in Fig. 50. Suppose the same requests are issued as shown in Fig. 49 (i.e., updates to LPNs 0, 0, 0, and 3), then one of the blocks from the write buffer is evicted to Log Block 0 based on the LRU policy. The BPLRU scheme reads LPN1 and LPN2 to pad the evicted block, and then performs sequential writes to LPNs 0, 1, 2, and 3 in Log Block 0. Afterward, Log Block 0 becomes Data Block 0 by performing a simple switch operation instead of an expensive full merge operation. Therefore, the BPLRU scheme significantly improves the random write performance.

The Effective Page Padding technique enhances the endurance of SSDs by reducing the number of erase operations [46]. This is achieved by employing the Evicted Block Table to store the block number and valid page numbers of the evicted block. Figs. 51 and 52 provide a comparison between BPLRU and the Effective Page Padding technique. Suppose write to LPN 0, write to LPN 2, the first buffer eviction, write to LPN 3, and the second buffer eviction occur in order. The BPLRU scheme reads LPNs 1 and 3 to perform page padding and then performs sequential writes to LPNs 0–3 due to the first buffer eviction as shown in Fig. 51. The write request to LPN 3
(D0) will be held in the write buffer. Finally, BPLRU performs page padding when the second buffer eviction occurs. Therefore, BPLRU requires two erase operations.

On the other hand, the Effective Page Padding scheme performs partial page padding up to the highest cached LPN as shown in Fig. 52. For example, if LPNs 0 and 2 are cached in the write buffer, the highest cached LPN is 2. Thus, LPN 1 is read from the flash memory to perform partial page padding because it is lower than the highest cached LPN. Since LPNs higher than the highest cached LPN in Data Block 0 (i.e., LPN 3 in Data Block 0) are not involved in page padding, they are instead stored in the Evicted Block Table. Then, the sequential write requests to LPNs 0–2 are issued to Log Block 0 during the first buffer eviction as shown in Fig. 52. Afterward, the write to LPN 3 is stored in the write buffer. When the second buffer eviction occurs, LPN 3 is written to Log Block 0. Therefore, the Effective Page Padding scheme requires only one erase operation. However, if the highest cached LPN is close to the block size, full page padding can result in better performance than partial page padding. The Effective Page Padding scheme defines a threshold for a padding management. If the highest cached LPN is lower than the threshold, partial page padding will be performed. Otherwise, full page padding will be performed, which is the same as BPLRU. As a result, the Effective Page Padding scheme reduces the number of erase operations compared to BPLRU by up to 18.2%.

Fig. 50 The page padding technique.
There is also a method called Predicted average Updated Distance Least Recently Used (PUD-LRU), which distinguishes the entries in the write buffer as Frequently Updated Group (FUG) and Infrequently Updated Group (IUG) to exploit temporal locality [56]. FUG and IUG are classified based on the update distance, which is similar to the update interval used in LAST (see Section 5.3.4). Moreover, the PUD-LRU scheme chooses a victim block among the blocks in IUG in an LRU manner. Thus, PUD-LRU considers both frequency and proximity of accesses.
11.1.8 Garbage collection aware striping

The Round-robin striping technique discussed in Section 9 is typically used to exploit channel-level concurrency, but it incurs extra garbage collection overhead. Garbage collection aware striping (GCAS) basically uses striping with block scheduling to improve garbage collection efficiency and thus performance [42]. Fig. 53 illustrates round-robin striping and GCAS. GCAS basically performs striping by grouping sequential logical pages in the same block.

Fig. 52 The Effective Page Padding method.

![Diagram of Effective Page Padding method](image)
Fig. 54 compares the garbage collection efficiency of the round-robin striping versus the GCAS. In this example, the SSD is assumed to have two channels each consisting of two blocks. When write requests occur to LPNs 0–7, the round-robin striping technique schedules writes to LPNs 0, 2, 4, and 6 to Block 0 and writes to LPNs 1, 3, 5, and 7 to Block 2 as shown in Fig. 54A. Afterward, another set of write requests to LPNs 0, 1, 2, and 3 causes LPNs 0 and 2 to be updated in Block 1 and LPNs 1 and 3 to be updated in Block 3. Since there are no free blocks, garbage collection is performed. Thus, LPNs 4 and 6 are copied to Block 1 and LPNs 5 and 7 are...
copied to Block 3, and then Blocks 0 and 2 are erased. As a result, the round-robin striping technique requires four copy and two erase operations. In contrast, the block striping technique allocates LPNs 0, 1, 2, and 3 to Block 0 and LPNs 4, 5, 6, and 7 to Block 1 as shown in Fig. 54B. When another set of write requests to LPNs 0–3 occurs, only one erase operation is needed leading to better garbage collection efficiency.

11.1.9 Harey Tortoise

The Harey Tortoise technique improves not only the best-case performance for latency-critical operations but also their average performance [48].
is achieved with a flexible FTL that considers the fact that access time for the first bit of MLC is much faster than the access time for the second bit of MLC. Therefore, the basic idea behind the Harey Tortoise technique is to map the first and the second bit of MLCs to fast pages and slow pages, respectively. The authors in [48] show that fast pages are 4.8 times faster than slow pages. Therefore, the Harey Tortoise technique provides fast pages for latency-critical operations and slow pages to maximize memory density, which provides similar performance to SLC with higher density using MLC device.

11.1.10 Willow
Willow is a flexible, user-programmable SSD that allows for a customized implementation to improve the performance. This includes complex atomic operation, native caching operation, delegating storage allocation decisions to the SSD, and offloading file system permission checks to hardware [44]. According to [57], a customized SSD that offloads file system permissions reduces latency by up to 58% and improves throughput by 7.6 times.

Fig. 55A shows the architecture of a conventional SSD with multiple CPUs and nonvolatile memory (NVM), i.e., flash memory, connected to the host system via the nonvolatile memory express (NVMe) interface. Although a conventional SSD has processing units, they are not directly programmable from the host system.

Fig. 55B depicts the architecture of Willow SSD, which has storage processor units (SPUs) each consisting of a microprocessor and a network

![Fig. 55 Architectures of a conventional SSD (A) and Willow SSD (B).]
interface to communicate with other SPUs. The host remote procedure call endpoints (HREs) are a set of objects that allow the operating system and applications to communicate with SPUs. Therefore, SSD applications can be installed on the Willow driver, and then it creates and manages HREs. This flexibility allows for supporting a variety of functions to improve their performance. For example, Willow provides an OS-bypass interface in order to avoid overhead from system calls and file systems, which allows applications to perform READ() and WRITE() operations without OS intervention. The performance of SSDs can also be improved by an OS-bypass interface when running trusted codes in the SSDs. Willow can also be programmed to be used as a cache for a larger conventional storage system. Furthermore, SPUs in the Willow can handle data intensive computations, thereby reducing expensive data movement from SSD to the host. These features allow for a user programmable system that can extend the behavior of an SSD and therefore improve its performance [44].

11.2 Lifetime and reliability

Since lifetime and reliability are important issues, there have been many research efforts that focus on improving longevity and reliability of flash memories. Lifetime and reliability are highly related because the lifetime of flash memories is determined by the reliability of flash cells. If a flash memory is no longer reliable, the FTL considers its lifetime as being over. Typically, wear leveling techniques are employed in order to improve lifetime. On the other hand, SSDs rely on ECC and Bad Block Management in order to improve their reliability. Bose–Chaudhuri–Hocquenghem (BCH) codes and low-density parity-check (LDPC) codes are widely used as ECC for SSDs [1]. Skip Block and Reserve Block methods are generally employed for bad block management (see Section 8) [7]. Sections 11.2.1–11.2.6 discuss some recent techniques to improve lifetime and reliability.

11.2.1 Dynamic program and erase scaling

There is a tradeoff between the endurance of flash memories and the performance of P/E operations. Erasing a NAND block with a lower voltage causes electrons to penetrate the oxide layer more slowly and gently, and as a result, there is less damage to the oxide layer than the traditional high-voltage method, which significantly improves endurance. However, the performance degrades due to slower P/E operations. The dynamic program and erase scaling (DPES) scheme exploits this tradeoff by dynamically choosing between a high voltage for a short P/E operation or a low voltage for a
long P/E operation [47]. DPES chooses the high voltage P/E operation when the number of outstanding write requests is high, and the low voltage P/E operation when the number of outstanding write requests is low.

11.2.2 Zombie NAND technique

The Zombie NAND technique exploits the gap in P/E cycles between TLC and MLC by resurrecting dead TLCs to MLCs. MLC and TLC have lifetimes of 10 k and 1 k P/E cycles, respectively. However, both MLC and TLC use exactly the same flash device structure except that TLC has more $V_t$ levels than MLC. Since TLC becomes unreliable after 1 k P/E cycles, the $V_t$ recognition process is changed to MLC allowing flash blocks to be used until 10 k P/E cycles. The Zombie NAND technique proposes a controlled wear-unleveling method as a way to resurrect dead blocks [17]. The wear-unleveling transforms TLC to MLC and MLC to SLC. Fig. 56 shows the advantage of the Zombie NAND technique. The capacity of TLC, MLC, and SLC decreases when the number of P/E cycles become above 1 k, 10 k, and 100 k, respectively. The Zombie NAND technique changes TLC to MLC after 1 k P/E cycles, and MLC to SLC after 10 k P/E cycles. Therefore, the resulting capacity follows the black line shown in Fig. 56, which extends the lifetime of flash memories. However, density is a tradeoff factor due to the reduction in the memory size.

11.2.3 Self-healing SSDs

As mentioned in Section 2.3, trapped charges inside the oxide layer increases $V_t$ of a flash cell. The self-healing SSDs technique recovers these damaged

![Fig. 56 Concept of Zombie NAND technique.](image-url)
cells using a high temperature to repel the trapped charges from the oxide layer [45, 58]. This is illustrated in Fig. 57, where an extra heater die is used to create heat. Therefore, the lifetime is improved by solving the endurance problem from trapped charges [45]. However, the extra heater die requires additional die area and power.

11.2.4 Flash correct and refresh
As mentioned in Section 2.3, charged electrons inside the floating gate can escape due to SILC causing retention errors. The flash correct-and-refresh (FCR) technique introduces a new method to improve lifetime, which is illustrated in Fig. 58 [49]. As time passes, electrons escape from the floating gate causing $V_t$ of the cell with data “0” becomes closer to the read reference voltage ($V_{READ}$). Therefore, the main idea of FCR is to periodically read and correct retention errors by performing in-place reprogramming. This is achieved by modeling the BER-based retention time for a damaged cell,

![Fig. 57 Trapped charges and self-healing SSD.](image1)

![Fig. 58 Concept of the FCR.](image2)
and then periodically refreshing the current data state “0”. This significantly improves the lifetime of flash memories and only requires modification to the SSD controller firmware. However, FCR requires constant power as well as extra processing to refresh flash cells. Furthermore, it cannot recover from retention failures that have already occurred during off power state.

### 11.2.5 Retention optimized reading and retention failure recovery

As discussed in Section 2.3, the state of data “1” cannot be properly read if the trapped charges causes $V_t$ to shift and becomes very close to $V_{READ}$. This issue can be resolved using the retention optimized reading (ROR) technique [50]. Fig. 59 shows $V_t$ as a function of the number of P/E cycles for MLC, which has three read reference voltages ($V_{READ1}$, $V_{READ2}$, and $V_{READ3}$). As can be seen, $V_t$ of erased data “11” (blue line) and $V_t$ of programmed data “01” (green line) increase as the number of P/E cycles increases. A read failure will occur when $V_t$ reaches points $A$, and thus the flash memory is no longer considered reliable. In order to avoid read failures, the ROR technique changes the read reference voltage $V_{READ3}$ to Optimized $V_{READ3}$ before $V_t$ reaches point $A$. Therefore, the lifetime of flash memories can be extended.

Even with ROR, the problem of charge leakage still exists. FCR discussed in Section 11.2.4 is one possible solution for SILC, but it cannot recover from retention errors that occurred during the power off state. The retention failure recovery (RFR) technique shown in Fig. 60 recovers data using a probabilistic method. This is done by classifying types of cells as fast leaking and slow leaking. In order to determine whether a cell is fast or slow.

![Fig. 59 $V_t$ as a function of P/E cycles.](image)
leaking, all the flash cells are programmed with the same voltage level. Then, after a certain amount time, fast leaking cells and slow leaking cells can be distinguished based on whether the remaining level is lower or higher than the mean threshold voltage level, respectively. This way, the original state can be predicted to recover the data from retention failures.

11.2.6 BER-based wear leveling and bad block management

The reliability and lifetime of flash memories are typically measured by the number of P/E cycles. However, flash cells do not become damaged uniformly based only on the number of P/E cycles, thus bit error rate (BER) is a more practical factor for properly reflecting the lifetime of flash cells. The BER-based wear leveling and bad block management technique introduces a BER-based damage estimator to more precisely measure the damage level of flash cells [33]. The BER-based wear leveling improves reliability by relying on the same BER threshold for all the blocks. In other words, some blocks that have exceeded the P/E cycle limit but have a good BER can be used until it dips below a threshold. However, extra processing is required to update BER for each block incurring a performance overhead.

12. Conclusion

This survey presented a comprehensive coverage of issues in SSDs, including all the basic functionalities provided by FTL. Moreover, it provided the state of the art in SSD research on improving performance, lifetime, and reliability. Our survey also clearly shows that the research challenges in SSDs are vast and many of the issues presented are intricately intertwined. Therefore, dealing with these issues holistically will be crucial for improving the performance of future SSD systems.
A comprehensive survey of issues in solid state drives

References


About the authors

**Youngbin Jin** received his B.S. degree in Electrical Engineering from Washington State University in 2010 and his M.S. degree in Electrical Engineering from the University of Texas at Dallas in 2013. He is currently pursuing a Ph.D. degree in Electrical and Computer Engineering at Oregon State University. His research involves cross-level issues between hardware and software and his interests include Computer Architecture, Computer Network, Multimedia, VLSI design, FPGAs, and Memory Systems.
Prof. Ben Lee received his B.E. degree in Electrical Engineering in 1984 from the Department of Electrical Engineering at State University of New York (SUNY) at Stony Brook, and his Ph.D. degree in Computer Engineering in 1991 from the Department of Electrical and Computer Engineering at the Pennsylvania State University. He received the Loyd Carter Award for Outstanding and Inspirational Teaching in 1994, the Alumni Professor Award for Outstanding Contribution to the College and the University from the OSU College of Engineering in 2005, and the HKN Innovation Teaching Award from Eta Kappa Nu, School of Electrical Engineering and Computer Science in 2008. He has been on the program committees and organizing committee for numerous international conferences including IEEE Consumer Communications & Networking Conference (CCNC), and IEEE International Conference on Pervasive Computing and Communications (PerCom), and IEEE Workshop on Pervasive Wireless Networking (PWN). He was the TPC-Chair for the 15th Annual IEEE Consumer Communications & Networking Conference (CCNC 2018). He was a Guest Editor for the Special Issue on “Wireless Networks and Pervasive Computing” for the Journal of Pervasive Computing and Communications (JPCC). He was also an invited speaker at the 2007 International Conference on Embedded Software and System and a Keynote Speaker at the 2014 ACM International Conference on Ubiquitous Information Management and Communication. He is currently the General Chair for the 17th Annual IEEE Consumer Communications & Networking Conference (CCNC 2020). He is also an Adjunct Faculty member at Korea Advanced Institute of Science and Technology (KAIST). His research interests include multimedia streaming, wireless networks, embedded systems, computer architecture, multithreading and thread-level speculation, and parallel and distributed systems.
CHAPTER TWO

Revisiting VM performance and optimization challenges for big data

Muhammad Ziad Nayyer\textsuperscript{a,b}, Imran Raza\textsuperscript{b}, Syed Asad Hussain\textsuperscript{b}

\textsuperscript{a}Department of Computer Science, GIFT University, Gujranwala, Pakistan
\textsuperscript{b}Department of Computer Science, Communication and Network Research Centre, COMSATS University Islamabad, Lahore, Pakistan

Contents

1. Introduction 72
2. Performance issues from big data perspective 74
   2.1 Single server issues 76
   2.2 Single data center issues 77
   2.3 Multiple geo-distributed data center issues 80
   2.4 VM performance overhead evaluation 82
3. Migration overhead mitigation techniques from big data perspective 83
   3.1 Single server techniques 83
   3.2 Single data center techniques 89
   3.3 Multiple geo-distributed data centers techniques from big data perspective 93
4. Summary, insights and open research challenges 98
   4.1 VM migration and big data 100
   4.2 VM storage migration and big data 101
   4.3 Effect of VM migration on collocated VMs with big data 102
   4.4 Cost effectiveness and big data 103
   4.5 Network traffic, heterogeneity and big data 103
   4.6 Hybrid of public and private clouds with big data 104
5. Conclusion 105
References 106
About the authors 111

Abstract

The concept of virtualization in cloud computing aims to maximize resource utilization and minimize cost by deploying multiple Virtual Machines (VMs) on a single physical server sharing resources such as CPU, Cache, I/O, and Memory. The sharing of these resources can cause severe performance degradation, thus requiring VM migration techniques for performance enhancement. The introduction of big data has made the performance enhancement more challenging due to extensive volume, velocity,
variety, variability, and veracity of data. The existing VM performance and optimization challenges need to be optimized for big data use cases. This chapter presents big data triggered VM performance challenges focusing big data applications and storage migration in cloud computing. State of the art VM migration techniques are evaluated against challenges posed by big data to outline possible solutions and research challenges.

1. Introduction

Big data refer to the volume of data that are challenging to handle due to its volume, velocity, variety, variability, and veracity. Frequency of the data generation has increased massively in the last decade demanding more computing resources to handle operations such as data capture, data storage, data analysis, and data visualization [1]. More computing resources mean more power utilization and complexity in terms of distributed algorithms and applications necessary to handle big data. Provisioning and handling such resources at the local level is costly and difficult. Therefore, organizations are seeking third party services. Cloud computing provides a viable solution for rendering infrastructure and resources by offering different service oriented model such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). Cloud computing make use of virtualization technology to maximize resource utilization with minimum cost. International Data Corporation (IDC) indicates that cloud computing will generate a revenue up to $53.1 billion by 2019 [2]. Cloud Service Providers (CSPs) use data centers to house cloud services and cloud-based resources. For cloud-hosting purposes, vendors often use multiple data centers in several geographic locations to ensure data availability during outages and other data center failures. However, there are various challenges faced by CSPs which include energy efficiency, seamless services, provisioning of the resource rich servers, cooling, high speed storage, and virtualization related challenges. Virtualization related challenges include rapid deployment, snapshotting, scheduling, prioritizing which may lead to resource contention. Provisioning more resources is a solution but not cost effective. VM migration on the other hand, is a more viable solution that is widely used. VM migration means that a VM facing performance issues due to limitation of resources (e.g., CPU, memory, storage, and network) is migrated from a single physical server to another resource rich server instead of provisioning more resource on the same server. This scaling technique requires less labor, time, and cost. In addition, it gives an
advantage of resource provisioning on runtime with minimum downtime. There are various methods of VM migration that are discussed in Section 3.2.

A VM usually comprise of disk image and memory state. However, in data centers, a central storage called Network Attached Storage (NAS) is used for storing data. As per volume of the data, a number of NAS devices may be used. Big data applications deployed on VMs are connected with the NAS devices over a high speed LAN. When a certain migration is triggered over the Wide Area Network (WAN) and a VM is migrated from one data center to another, it’s affiliated storage must also be migrated with the VM. Migration of affiliated storage is essential for successful working of the migrated VM in the new environment and is a major challenge as the volume of the data is very large. Efficient algorithms are required to reduce data size, network traffic, and demand for bandwidth while preserving energy [3,4]. Additionally, a fast, reliable, and secure network is also required for the migration of NAS. Hence, big data networks are the only solution to survive the challenges of future Internet [5]. The term “Big Data Network” is relatively new and being used rarely in the literature. However, it refers to the readiness of networks for big data [6–13]. Many big data network architectures have been proposed based on Software Defined Networks (SDNs) and Hadoop by different vendors such as Juniper [14], IBM [15], CISCO [16], and Oracle [17]. SDNs employ the abstraction of lower level functionality that separates the control and data plane to achieve both the programmability and speed of data transfer [18]. Hadoop is a platform providing structure for big data that offers data and content management along with warehousing functionalities [19], but state of the art virtualization techniques is yet to be equipped with Hadoop and SDN solutions.

It is imperative to analyze the existing techniques used for VM performance enhancement from big data perspective. We have classified hardware parameters and operational issues effecting the VM performance. This will help us to focus on the major points for the improvement of existing techniques as well as for the development of new ones. The contribution of this study can be segregated in two parts: In the first part, we have assessed performance enhancement techniques according to the classified performance parameters and operational challenges. In the second part, we have analyzed the same performance enhancement techniques from big data perspective and also proposed a future cloud model ready to handle big data challenges.

Rest of the chapter is organized as follows: Conventional VM performance issues in a single server, single data center and geographically
distributed data center from big data perspective have been discussed in Section 2. Existing overhead mitigation techniques and their tradeoffs have been presented for big data in Section 3, future cloud model, insights and open research challenges have been mentioned in Section 4, and finally Section 5 concludes the chapter.

2. Performance issues from big data perspective

The world has moved from single server virtualization to single large data center and to geo-distributed data center. We have categorized the virtualization challenges from these three perspectives as shown in Fig. 1.

However, there are some common operational challenges as shown in Table 1.

Detail of performance challenges is as follows:

1) VM performance can be limited by poor prioritizing. Issues arise when the element of priority is not considered while performing resource allocation, thus increasing the risk of performance overhead [20].

2) Scanning for viruses and malware on host or guest OS also causes performance degradation [21,22] as it is a CPU intensive task. The duration of resource occupation is highly dependent upon the size of host and guest OS along with the data.

3) Some operating systems (OS) do offer power saving option which reduces the processor clock speed causing performance issues [21].

4) Migration operation brings substantial overhead on collocated VMs at source and destination servers as it utilizes computing and network resources. A poorly scheduled migration operation can severely degrade the performance. Concurrent migrations may also bring significant performance overhead [3].

Fig. 1 Consolidated view of VM performance overheads.
<table>
<thead>
<tr>
<th>Formation</th>
<th>Prioritizing</th>
<th>Antivirus/anti-malware scanning</th>
<th>Power saving</th>
<th>Scheduling</th>
<th>VM migration</th>
<th>Storage migration</th>
<th>Rapid deployment</th>
<th>Rapid snapshotting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single server</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Single data centre</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Multiple geo-distributed data centre</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Timing and scheduling are two other factors that must be considered for operations such as VM snapshotting and migration. A poorly scheduled snapshotting or migration task may not only affect the VM but also severely affects the collocated VMs [21].

The host operating system’s performance also affects the VM. Disk speed, cache, and bus are included in those factors effected by host OS. In Redundant Array of Independent Drive (RAID), if the host OS and VMs are on the same spindle then disk cache and bus sharing can cause significant overhead to the VM [23]. If a server gets low on memory, the operating system makes heavy use of virtual memory and the constant paging operation cause tremendous load on disk I/O resulting in performance degradation [24].

2.1 Single server issues

The basic goal of virtualization is to multiplex physical server’s resources such as CPU, memory, and storage to provide better resource utilization as shown in Fig. 2.

The efforts for single server virtualization are focused on the isolation of shared resources such as CPU cache, control and data bus, memory bandwidth, disk I/O, and disk I/O buffer which are the major cause of contention, resulting in a performance overhead [25]. There are various hypervisors and VM monitors such as Citrix XenServer, VMware vSphere, and Microsoft Hyper-V [26]. The hypervisors are proven to be good in providing
resource isolation for multiple VMs. Isolation is achieved through full fledge CPU schedulers (e.g., Credit Scheduler in Xen [27]) and use static disk partitions. Routine operations such as rapid deployment and snapshotting which are very vital for the smooth execution of a VM are also a source of performance overhead. However, the effects of deployment and snapshotting are restricted within the boundary of a single server. The impact of all aforementioned factors is also linked with the nature of the workload. Workloads can be classified on the basis of resource intensiveness in terms of CPU, I/O, and memory, etc. In the presence of big data, the problems induced by the shared resources are increased drastically. Timely completion of larger jobs having increased computations and read/write operations require more supporting hardware in the form of CPU, memory, I/O, bus speed, bandwidth, and cache, etc.

2.2 Single data center issues

In the single data center, VMs are migrated over Local Area Network (LAN) from one server to another for performance improvement as shown in Fig. 3. The migration operation effects the performance of collocated VMs on both source and destination servers due to resource contention among migrated and already deployed VMs.

When a large volume of data is to be processed for complex jobs, it is highly desired by customers to lease a group of VMs from cloud service providers. MapReduce [28] is an example of such a complex job that requires extraordinary resources to run distributed algorithms using multiple VMs within a single data center. A single VM migration from the leased group of VMs can affect the overall performance due to the dependency among group VMs. Other examples may include scientific analysis and simulations that require more than a single VM’s resources. From aforementioned
examples, the question arises that whether the techniques used to reduce performance overhead for a single server can also work for a single data center? The answer to this question is “No” as involvement of LAN and requirement for VM migration for load balancing does not persist for a single server. Big data demands for more resources in terms of storage, network processing, and bandwidth. A big data flow can affect the performance of a data center as it requires more network bandwidth and may trigger additional migrations for the handling and management of big data. Hence, the impact of big data challenges is more obvious in a data center environment than for a single server. We have discussed single data center issues such as shared resources, migration effects on collocated VMs, rapid deployment, and snapshotting in this section as shown in Fig. 4.

2.2.1 Migration

VM migration technique is commonly used for saving power and managing load in a cloud data center, but incurs a huge performance overhead which is unavoidable especially in case of concurrent live migrations [29]. The volume of data and supporting infrastructure is also an issue if speedy migration is required. The effects of concurrent deployments can also be seen on source and destination servers, including the applications running on the migrating VM itself. Migration and deployment operations further limits the CPU and network resources resulting in a longer migration time [30]. In LAN, where storage is not migrated as shared storage devices are used in the form

Fig. 4 Single and geo-distributed data center issues.
of independent NAS device, storage is not considered a challenging factor from big data point of view. However, active connections are closed and re-established from new destination. If a data intensive application having greater number of read and write operations is running on a VM that is in the middle of a migration process, then termination and re-establishment of active connections will result in a service disruption. There are two commonly used migration types, cold migration and live migration. In cold migration, a powered off VM and in live migration, a running VM is transferred from the source to destination server. Both techniques have their own pros and cons. In cold migration, the services running on the server are suspended during migration. The VM is first powered off and then migrated. The VM needs to be powered on after migration on the destination server which also takes some time lingering on the services running on the VM. In live migration, the static part (not changing or having any active write operation) of the VM is transferred first while VM keeps running on the source server. Then VM is suspended and the remaining part is transferred. Live VM migration reduces the down time but engages more resources for a longer time. Existing solutions of VM migration have different tradeoffs such as interference of live migration on collocated VMs against more CPU utilization [3], uninterrupted services during migration against more network and CPU load [3], and performance against more energy requirement [4]. There are many variants of both techniques which are further discussed in Section 3. All approaches try to optimize performance and achieve maximum utilization.

2.2.2 Deployment and snapshotting
Concurrent deployment and snapshotting are two consistently performed operations in a data center [31]. However, the nature of deployment and snapshotting operations in a data center is different from a single server. For the execution of complex and lengthy jobs having a large volume of data to be transmitted, it is very vital for such jobs to be timely executed using the fast deployment method. The fast deployment method requires a number of VMs to be initialized concurrently in cloud environment using more than one physical server [32]. The initialized VMs altogether work as a cluster and their combined resources are used to execute complex jobs. The cluster of VMs requires a tight coupling of resources through LAN and may use shared storage. VM snapshots are also maintained on the shared storage thus increasing the network dependency. Performance of a single VM on a physical server belonging to a cluster affects the other VMs on different physical servers belonging to the same cluster due to dependency of computations.
The main reason for the performance overhead is the large volume of data that are generated during rapid deployment and snapshotting operations. This big data result in a massive amount of network traffic interfering with regular data center traffic.

2.2.3 Shared resources
In the cloud environment, the services are provided in a shared model. Resources such as CPU cache, disk I/O and buffer, network I/O and memory bandwidth are shared among VMs. However, nature of these shared resources in a data center is changed as compared to a single server. The most critical shared resource in a data center is network bandwidth and I/O. The challenge of network bandwidth and I/O contention arises mostly at network storage level due to multiple VMs sending requests simultaneously. The I/O contention is dependent upon the volume of data coming in and going out of the storage device. In case of big data, where data are stored on multiple shared devices, the I/O contention will be at every device level resulting in a delayed operation. The latest statistics by Amazon EC2 confirm that the throughput variation faced by Standard Medium Instances (SMIs) can be up to 65% of the total network I/O [33], and the write I/O bandwidth can vary by as much as 50% from the mean [34].

2.3 Multiple geo-distributed data center issues
All issues discussed for the single data center are also valid for geo-distributed data centers. However, migration of VMs in a geo-distributed data center is treated differently due to the involvement of WAN. In the WAN environment, not only VM’s disk and memory state is transferred but affiliated storage is also transferred [35]. The task of transferring affiliated storage becomes difficult due to the challenges of handling large data size, network traffic, link condition, available bandwidth, downtime of the VM in migration process, and energy consumption. An example depicting VM migration in WAN environment has been produced in Fig. 5.

The aforementioned challenges not only affect the source and destination data centers, but also the migrating VM itself and services running on it. Solutions for the single server or single data center do not work for geo-distributed data center environment due to the increased network volatility. The study in Ref. [36] has mentioned an experimental migration performed over WAN between the data centers in Illinois and Texas. The link was configured with a maximum throughput of 85 Mbps and VM was configured with 1.7 GB memory and 10 GB disk size. A web application
was hosted on the VM. The experimental results showed that it took around 40 min for disk transfer and 210 ms for memory transfer. During the migration process, the response time of web services was degraded to 52 ms on average, whereas, it was recorded as 10 ms before migration. The migration time increases with the increase in disk size. However, no storage was migrated in the experimental migration. In the presence of big data where a VM itself may consist of several disks due to distributed applications and algorithms running on it and have a number of affiliated storage drives, the impact on time will be drastic.

There is no doubt that the live migration technique used for VMs to move from one data center to another is very useful in reducing the services downtime, but it suffers from latency and bandwidth issues. However, big enterprises and IT firms are getting benefits from VM migration technique to transfer a single or number of VMs from a heavy-loaded data center to a light-loaded data center. The purpose of VM migration is to balance the load, save energy and enhance the resource utilization of the light loaded data center. The same technique is also used to move a single or number of VMs to a more powerful data center having high computing resources when the resource need of a VM is not being fulfilled. The cloud “Bursting” technique gives opportunity for small scale business companies and enterprises to jump between private and public clouds whenever their resource need is increased and are charged only for that period [36].

Table 2 summarizes all the major factors effecting the performance of single server, single data center and geo-distributed center.
Two types of metrics are used for the evaluation of VM performance overhead. First is performance degradation metric, which can be calculated by comparing the performance degradation of various applications running in an IaaS cloud with the application running in an isolated environment.

\[
\text{Performance Degradation} = \frac{\text{VM(IaaS)} - \text{VM(Isolation)}}{\text{VM(Isolation)}}
\]  

Where, VM(IaaS) is VM in an IaaS cloud and VM(Isolation) is VM in an isolated environment. The larger magnitude of performance degradation indicates severity of the VM performance overhead. The second metric represents the variation in VM performance over an interval of \(\Delta t\). VM performance variation can be represented by the coefficient of variation [37], and is formulated as

\[
P_{v\Delta t} = \frac{1}{\bar{x}} \sqrt{\frac{1}{n_x - 1} \sum_{i=1}^{n_x} (x_i - \bar{x})^2}
\]

where, \(x_i\) represents performance measured for VM, \(P_{v\Delta t}\) represents variation in VM performance, \(n_x\) represents the number of performance measurements taken and \(\bar{x}\) represents average VM performance over time \(\Delta t\). The magnitude of performance variation directly corresponds to server’s performance overhead. It is very difficult to get the actual performance degradation measures due to security and measurement cost involved. Probing hardware and software requires additional tools and permission to extract performance parameters. When both discussed metrics fail to provide an accurate measure due to unavailability of tools to communicate with the

<table>
<thead>
<tr>
<th>Configurations</th>
<th>Shared resources</th>
<th>Deployment</th>
<th>Snapshot</th>
<th>VM migration</th>
<th>Network latency</th>
<th>Storage migration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single server</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Single data centre</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Multiple geo-distributed data centre</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 2: Virtual Machine performance overhead factors for different configurations.
hardware and software at the lower level, a third type of metric is used to measure the performance overhead, which considers the performance measurement of routine operations during live migration such as the amount of data to be transmitted, network traffic, link and bandwidth condition, downtime of the VM being recently migrated, and energy consumption.

3. Migration overhead mitigation techniques from big data perspective

Numerous approaches to minimize the VM performance overhead and optimize the recourses in a cloud environment have been proposed. These overhead mitigation and optimization approaches have been categorized according to single-server, single data center, and geo-distributed data center, respectively, and their overview have been provided in this chapter from big data perspective.

3.1 Single server techniques

Recent hypervisors such as Microsoft Hyper-V and XenServer [34] provide isolation for CPU, memory, and disk resources, but lack the capability of isolating the resources such as CPU cache, network I/O, memory bandwidth, and disk I/O. Many techniques have been proposed to overcome the resource contention problem and minimize the VM performance overhead on a single server. We further divide these techniques in two categories: First is based on collocated VM resource isolation and second is based on VM assignment optimization.

3.1.1 Collocated VM resource isolation

Resource isolation is a very common technique to avoid resource contention among collocated VMs on a single server. However, the resources that are hard to isolate such as CPU cache and memory bandwidth can also be handled to some extent using techniques such as Q-Cloud [38]. For better understanding, we have categorized the resource isolation approaches according to targeted resources of CPU, memory, cache, and I/O.

3.1.1.1 CPU targeted approaches

Q-Cloud uses an online feedback mechanism to build a model using Multi Input Multi Output (MIMO) to capture the performance interference interactions. The captured interference information is further used to perform closed loop resource management, which dynamically identifies
the underutilized CPU resources. Application’s resource need is provided by workload SLA and then required resources are dynamically provisioned. An application specific metric is defined in the SLA and semantic-less feedback signals are used for the online adaptation. A structural view of Q-Cloud is shown in Fig. 6. When we analyze aforementioned approach from the big data point of view, we can see that the online feedback consumes much bandwidth and suffers from latency issues considering the large volume of data. The frequency of performance interference interactions will also be increased resulting in delayed provisioning of resources making Q-Cloud technique unsuitable for big data applications.

3.1.1.2 Cache targeted approaches
Partitioning the cache space is a solution to overcome the contention issues among cache resources. It uses a coloring scheme to assign each virtual and physical memory page a different color. Only a same colored virtual page can be assigned to a physical page [39]. In the presence of big data, the process of partitioning cache space will become difficult as cache space is very limited and with the increase of data variation, the frequency of cache change will also be increased. Hence, this will result in increased frequency of mapping between virtual pages and physical memory pages consuming more CPU cycles causing an overhead.

A copy-based cache service is being introduced in Ref. [40] that improves the read performance by conserving a working set in shared
memory with low management cost. There are two conditions for the cooperative caching scheme that system must use shared storage for all clients and cooperative cache hit rate must be greater than disk access. The cooperative cache scheme has been implemented using Xen with split virtual drivers (front-end and back-end) arrangement for transparent I/O virtualization. Conditions required for the cooperative cache scheme will not hold for big data as cache hit rate will decrease and disk access will increase due to increased data volume, variety, and variability, resulting in poor performance.

3.1.1.3 I/O targeted approaches

Disk I/O contention is also a problem for the single server virtualization. One solution is to force the disk I/O scheduling on VM and Virtual Machine Monitor (VMM) levels. The selection of optimal disk I/O scheduler for collocated VMs can significantly improve disk I/O resources [41]. However, the tradeoff between disk I/O fairness and performance of collocated VMs is not very flexible and can be achieved at the cost of I/O throughput and latency. Big data either require multiple disks or separate NAS devices for storage. In both cases I/O scheduling at multiple levels is required to avoid disk I/O contention as a whole. So, the disk I/O scheduling only at VM level or VMM level will not work for big data.

A priority based scheduling technique has been proposed in Ref. [42] that improves the response time of I/O requests. In order to improve the response time an I/O request is interrupted at VMM level and the requests are reordered on physical server in the disk I/O queue according to priority. The network bandwidth problem can be solved through hardware solutions such as attaching multiple network adapters or using multi-queue based network adapters. It can also be solved through software solutions such as bandwidth capping, where a bandwidth cap is set in the VM configuration before booting [34]. For big data having increased number of I/O requests two problems will occur: First, a very long queue will be required and second, the priority queue’s efficiency decreases with the increase in length as less priority tasks face longer delays.

In Nicpic [43], the state management packet scheduling task responsibility has been separated from CPU. The tasks such as classification of packets, putting them in queue, and application of rate limits are performed by CPU. The packets are queued in memory and scheduled for extraction according to the rate limit of VMs. Both scheduling and extraction tasks are performed by Network Identification Card (NIC) and direct memory access mode is used to extract the packets from queue to NIC. A very long queue is
required to handle big data resulting in more memory requirement at NIC level. Since NICs are not capable of handling complex computations, the scheduling and extraction of large number of data packets will become difficult and time consuming resulting into delayed services.

A packet aggregation based mechanism overcomes the memory latency problem caused by network I/O while transferring packets from driver domain to guest OS [44]. Driver domain is used to provide access to shared network devices. One large packet is created by combining several small packets and transferred to the aggregation destination. More data can be transferred with aggregation mechanism since there will be less number of memory requests and revocations, fewer copies, and reduced notifications. Due to big data, the frequency of requests to driver domain at aggregation destination will increase. The reversal of the aggregation process requires extraction of small packets incurring more CPU load, hence resulting in overall performance degradation.

3.1.1.4 Memory targeted approaches

A Xen balloon driver [45] based framework has been proposed to control memory transaction upon the joining of multiple VMs [46]. This framework has been implemented in user space to avoid virtual machine monitor interference. A global-scheduling algorithm solves linear equations, achieves a global solution that adjusts itself according to the availability of resources using dynamic baselines. A working model of the discussed framework has been shown in Fig. 7. In the presence of big data where

![Fig. 7 Automatic memory control system.](image-url)
multiple VMs, distributed algorithms, and storage drives are employed for a single larger task, the existence of a global-scheduling algorithm is difficult to realize.

### 3.1.2 VM assignment and placement optimization

The second approach to solve the problem of VM performance overhead is to find an optimal mapping between VMs and physical servers. The techniques presented in Refs. [47,48] propose a prediction based algorithm using a combination of weighted mean with the linear and non-linear model, respectively. Statistical machine learning has been used for reasoning and discovery of optimal mapping. Based on the predicted data, an interference aware scheduler is employed that ultimately reduces performance interference. However, due to high volume, velocity, variety, variability, and veracity of the data, an accurate prediction is challenging.

A cache cloning scheme has been introduced in Ref. [49] for better VM placement. The cache usage estimation is performed through active probing and used for the prediction of performance degradation of an application when placed with other applications. A synthetic cache loader benchmark is used to create a clone for every application arriving at the host platform. The purpose of an application’s cache clone is to mimic its cache pressure, which includes both the cache space and the memory bandwidth occupancy of the application. The cache clones are later used as a proxy for actual applications while predicting performance degradation in co-location scenarios. As already discussed, due to high volume, velocity, variety, variability, and veracity of the data, an accurate prediction is hard to make. On the other hand, probing itself becomes a problem when a large volume of data traffic is to be analyzed.

A mechanism to improve VM performance using pSciMapper framework has been discussed in Ref. [50]. The proposed scheme helps in establishing the requirement of physical servers by assigning workflow and VM to each server. A dimensionality reduction technique using Kernel Canonical Correlation Analysis (KCCA) is used to connect the key features with power consumption and time of execution. Hierarchical clustering and the Nelder-Mead optimization algorithm are used to search for optimal consolidation. However, involvement of big data creates a problem for the discussed approach as increased variability of the workflow directly affects the requirement of physical servers and choice of VM, making it difficult to find an optimal consolidation.
A Last Level Cache (LLC) based approach has been presented in Ref. [51] to overcome the VM resource contention. The VMs on physical servers are tracked down having maximum and minimum LLC miss rate. VM having maximum LLC miss rate is swapped with the VM having minimum LLC miss rate. Since in the presence of big data, the cache change will be very rapid and the frequency of LLC miss will be increased posing a greater overhead to the performance of LLC based approach. However, the threshold calculations for optimal performance are not available to evaluate this approach more accurately for big data.

The approach with qualitative prospect in Ref. [25] to identifies resource contention through a clustering technique of low level metrics called DeepDrive. A metric is built by observing application behavior containing information about performance counters and statistics received from hypervisor. Main components of DeepDrive have been shown in Fig. 8.

DeepDrive uses two methods of interference analysis. The results of accuracy and overhead calculations differ for both methods. A warning system is being deployed on VMM which provides early interference statistics. These interference statistics are placed in a multi-dimensional space where both interference and non-interference cases are placed in separate clusters. However, the processing involved for interference analysis increases with the arrival of big data and behavioral analysis becomes very hard due to the requirement of larger datasets analysis.

Xu et al. proposed a technique in which they first identify latency sensitive applications and transfer them to a well-resourced VM for performance improvement [52]. The focus of this study is on the tail of round-trip latency as it produces diversified impact on user experience. An example of

---

![Fig. 8 DeepDive overview, how it detects and mitigates the effect of interference.](image)
the Amazon’s EC2 has been used and the analysis show that the induced latency is due to the poor response time of nodes rather than the network. Applications handling big data are even more resource hungry than latency sensitive applications, but it has not been considered in the proposed scheme. In addition, the element of round-trip latency is not a major problem for data intensive applications.

The discussion about single server techniques show that existing techniques are unable to support big data applications. A comparative analysis of all aforementioned techniques from big data perspective has been presented in Table 3.

3.2 Single data center techniques

The performance of VMs can be degraded due to routine operations such as concurrent deployment, migration, and snapshotting. However, major focus from single data center point of view is on the migration operation as it brings significant overhead [53,54]. Migration in a single data center is performed over LAN. We can further divide VM migration techniques for the single data center in two categories for better understanding: First is based on VM and migration destination selection and second is based on data size and network traffic. We have analyzed VM migration techniques for the single data center from big data point of view.

3.2.1 VM and migration destination selection

Liu et al. [55] has developed a cost model based on performance and energy modeling to calculate the cost of each VM candidate for migration. VM candidate having the least cost is selected for migration. An application-oblivious model has been devised that uses learned knowledge about the workload at VMM level to predict the cost. However, workload analysis of a big data job at VMM level can cause increased number of interferences at VMM level resulting in poor performance. When distributed storage drives are employed, the knowledge at VMM level is not enough to accurately calculate the cost of a VM.

The scheme in Ref. [56] observes the VM migration effects on three entities, i.e., migrating VM, collocated VMs, and applications running on them. The migration operation consumes a lot of CPU cycles and hence affects the performance of both source and destination servers. The concept of Service Level Objective (SLO) penalty is a key consideration of the proposed approach. The VM with the smallest SLO penalty is selected for migration. Since SLO penalty is measured offline at VM level, the
<table>
<thead>
<tr>
<th>Techniques</th>
<th>CPU cache</th>
<th>Memory</th>
<th>Memory B/W</th>
<th>Disk space</th>
<th>Disk I/O</th>
<th>Disk cache/Buffer</th>
<th>Network I/O</th>
<th>Network B/W</th>
<th>Control bus</th>
<th>Data bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU core compensation [38]</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Hot page coloring [39]</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>VMM-and VM-Level I/O schedulers [41]</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Priority-base I/O schedulers [42]</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Rate limiting [43]</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Copy based cache service [40]</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Packet aggregation-based mechanism [44]</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Automatic memory control [46]</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Interference modeling [47]</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
</tr>
<tr>
<td>Interference calculation [48]</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
</tr>
<tr>
<td>Cache clone [49]</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Metrics correlation [50]</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Cache partitioning [51]</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>VM assignment [52]</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
NAS placed outside the VM is ignored thus rendering the SLO measure inaccurate. Big data storage drives attached to VMs must be considered while measuring SLO penalty. Neglect of the element of storage renders the proposed approach unsuitable for big data applications.

The idea based on a combination of Ant Colony Optimization (ACO) and 2-opt local search algorithms optimally places the VM. This reduces the maximum link utilization [57]. The traffic statistics are obtained from a hypervisor and proposed algorithms are applied for obtaining the optimal candidate to place the VM. However, only traffic statistics at hypervisor level are not enough for big data handling and management.

iAware [58] uses an analytical model to calculate the effects of VM migration. The VM with least effecting factor is chosen for migration. The calculation for the least effecting VM is performed online using a multi-source demand and supply model. Only the effects of VM are calculated and the decision is solely based on this calculation. Big data requires that the impact of other parameters such as volume, velocity, variety, and veracity are considered.

### 3.2.2 Data size and network traffic optimization

A technique based on pipeline parallelism has been proposed in Ref. [59]. The proposed technique tries to balance out both the processes and data required for migration operation by using the idle CPU and network resources. This helps to expedite the parallelization opportunities available for the live VM migration. However, the scope of the proposed technique is limited to VM level. Distributed data storage drives and impact of big data challenges has not been addressed.

There are many other approaches to overcome data size and network traffic problem, one such example is transfer of data during VM migration by reducing its size. This problem can be mitigated by using any data compression technique like MECOM [60] but will definitely require more CPU cycles. Since CPU is not considered to be a major problem in VM performance, so, one can get the benefit by shifting the network load to CPU. But how much load should be shifted is still a question? The same compression algorithm should be implemented on the destination server to decompress the compressed contents and restore memory state from it. The majority of compression based techniques are only applied to VM and memory. VM and memory are very less in size compared to the affiliated big data storage drives. Both compression and decompression tasks consume large amount of CPU cycles, and for big data the compression and decompression operations can
cause CPU resource contention and delay due to the processing of large volume of data. Hence, the proposed techniques to reduce data size and network traffic are inadequate for migrating big data storage and applications.

There are some techniques that reduce the network traffic during live VM migration. Each technique uses a different method to achieve the same goal (e.g., [61–64]) of reducing network traffic. Delta Compression Technique (DCT) also reduces the network traffic by transferring only the changed data between the current and previously transferred data instead of transferring the whole image or disk block in each round of pre-copy. DCT is particularly suited for an application having more memory dirty rate and slow network connection. Compression and decompression operations consume a lot of CPU cycles and cause delay when a large volume of data is used. Further, identification of changed data from a very large dataset is also a CPU intensive task adding more intensiveness to the CPU resource.

On the other hand, the memory pruning [61] or de-duplication method [62,63] transfers only the critical data of OS kernel and application, which is necessary to run the VM and the application. The workflow of Generalized Memory De-duplication (GMD) engine has two stages. The Introspection stage identifies free memory pages in guest VMs and the de-duplication stage duplicates pages using hashing and byte-by-byte comparison. However, hashing and byte-by-byte comparison for the big data requires more CPU cycles and time. If a big data application is running on the VM in migration process, handling the critical data of such an application is not a problem rather than handling and management of the associated big data flows.

CR/RT-Motion [64] uses a log based scheme. It logs events, and when migration is required, these logs are only transferred to the destination server and replayed to restore the memory state. Logging each and every event for a large volume of data may consume more CPU cycles. The concept has not been tested from the big data point of view. However, it will require more computing resources.

In Jo et al.’s technique [65], only those memory pages are transferred that are not available on shared storage. The information of memory location and list of storage blocks is also transferred to the destination server. This information is used to restore the memory state of the migrated VM on the destination server. If we consider the same mechanism for big data storage drives, we can observe that finding available contents and their memory locations from a large volume of data will be difficult to handle. Further, compiling the list and transferring storage blocks for such large and rapidly
changing dataset will have an impact on CPU and bandwidth consumption which has not been considered in discussed technique.

Some more schemes focusing on the reduction of network traffic include live gang migration [29]. The live gang migration scheme proposes that same operating systems, applications, and libraries can have a major amount of similar memory contents. Hence, the objective is to locate similar memory contents and transfer a single copy resulting in a reduced amount of network traffic and data. The method of de-duplication [62,63] technique is widely used to find the same memory contents in VM memory and disk image. Afterward, only a single copy of the contents is transferred. The concept of finding similar contents is a successful factor for VM memory, but there is a less chance to find the similar contents on storage. Hence, storage migration is still an unaddressed challenge.

Breitgand et al. [66] has introduced a mathematical model to learn the tradeoff between the VM performance and migration time. The results obtained from the mathematical model is used to allocate a suitable amount of bandwidth for live migration. When same number of VMs are considered for migration and two different techniques are used, it has been observed that sequential transfer of the VMs causes less overhead than a concurrent live migration [67]. The scope of the proposed model is limited to VM only. Affiliated storage and its migration has not been considered.

The significance of the Named Data Networking (NDN) technique [68] is that it uses names of the services to identify the communication with a VM, while other techniques use network addresses. Routing is also performed via service names which results in reduced migration and downtime, and also achieve location independent environment for services and VM. The proposed technique can be employed for big data applications, but no testing has been performed from big data perspective. A comparative analysis of all aforementioned techniques from big data perspective has been provided in Table 4.

3.3 Multiple geo-distributed data centers techniques from big data perspective

The VM migration techniques used for geo-distributed data centers differ from the single data center due to the involvement of WAN. Similar to the single data center platform, we divide geo-distributed data center approaches in two categories: First is based on VM and migration destination selection and second is based on data size and network traffic. However, more focus is on the latter category as in geo-distributed data centers, greater
Table 4 Comparison of approaches to improve VM performance overhead for single data center.

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Migration time</th>
<th>Migration B/W</th>
<th>Migration traffic</th>
<th>Migration size</th>
<th>Migration speeding</th>
<th>Migration cost</th>
<th>Application down time</th>
<th>Application ready</th>
<th>Big data ready</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block comparison [29]</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Cost metrics [55]</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>SLO penalty [56]</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>VMP scheme [57]</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Least performance interference [58]</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Parallelizing migration [59]</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Compression [60]</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Filter useless memory pages [61]</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Compression [62]</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>De-duplication [63]</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Transfer and replay of execution log [64]</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Transfer location of memory [65]</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Mathematical modeling [66]</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Sequential transfer [67]</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Named data networking [67]</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
challenges are limited bandwidth and latency over WAN. Especially when big data is employed, the challenges of limited bandwidth and latency become more severe.

### 3.3.1 VM and migration destination selection

The technique proposed in Ref. [69] addresses the correlated VM problem which is due to the distribution of multi-tier application over different VMs. All correlated VMs should be transferred to the new destination for proper functioning of the multi-tiered application. A coordinated system called VM buddies has been devised that uses a synchronization protocol to ensure the concurrent migration of all correlated VMs. An adaptive network bandwidth allocation algorithm and a costing model based on pre-copying is used. However, the proposed technique produces a large volume of data in the form of VM disk images and storage drives. Big data in the form of distributed storage drives must also be transferred along with VM disk images, but the proposed technique lacks in providing a solution for storage migration. If relevant big data is not transferred and a link to distributed storage drives is made, then network latency over WAN affects the performance.

Another solution for the problem of correlated VM problem has been presented in Ref. [70]. Following condition is imposed in the proposed solution: Correlated VMs should belong to a single Virtual Data Center (VDC) and to a single VDC request. However, the request may consist of virtual resources from multiple VDCs.

### 3.3.2 Data size and network traffic optimization

The first attempt to transfer a running web server over a WAN environment has been made by Bradford et al. [36]. In order to complete the migration process, an iterative loop was created through which the memory state and disk image of the VM was transferred. There is a bottleneck in the proposed migration process, when write intensive workloads are to be transferred, loop iterations are drastically increased. However, the problem of increased iterations has been addressed through write throttling mechanism that limits the number of write operations to control the transfer and disk dirty rate. The current network connections are dynamically redirected to the destination server using IP tunneling technique. However, the proposed technique does not cover the migration of affiliated storage drives incorporating big data. Incase big data is considered; the write throttling technique will create a bottleneck due to increased number of write requests. This results in a very long iterative loop which not only consumes more CPU cycles but network
and bandwidth resources as well. The complete process will also take more time than expected, which is not desired.

The mechanism in Ref. [36] reduces the number of dirty pages and rounds of memory pre-copy phase by transferring only the difference between current and previously transferred pages. The cost of VM migration is higher than the memory state transfer. The migration cost is reduced by replicating the VM images in the background before live migration, and then synchronizing the disk states later [71]. However, comparison of current and previously transferred pages involving big data will take more time due to the increased number of comparisons. The incremental effect in comparisons will also bring a heavy load on CPU effecting the performance and making the proposed approach unsuitable for big data applications.

Another technique proposed by Akiyama et al. [71] is for the data intensive applications that uses page cache teleportation for the reduction of data size to be transferred. The proposed technique uses memory addresses to successfully migrate the VM to the destination server and the process of memory location identification is performed before the actual migration is in the process. The whole operation is performed using the following steps: (1) The disk image is synchronized before a migration. (2) Write access to the memory are tracked during the following steps. (3) The kernel module detects where the restorable page cache is on the guest memory and the disk. (4) The module sends them to the VMMs. (5) The destination VMM copies the disk blocks to the guest memory. (6) The source VMM sends memory pages except the restorable page cache, which has not been modified during the steps (3)–(5). Storage migration is not covered in the proposed technique. However, if we analyze the proposed technique from big data storage perspective, we can say that larger storage size will take more time to synchronize. Tracking a large number of write operations and locating unmodified and restorable pages will definitely consume more CPU cycles making the proposed technique resource intensive, costly, and undesirable.

A prioritization based technique has been proposed in Ref. [72] to transfer VM data. The proposed method prioritizes the transfer of critical data that are the data required for booting the OS and running the user application. However, the proposed technique lacks in providing the solution for the non-critical data. For a big data application, critical data are not a problem, but big data flow is a major challenge that has not been considered in the proposed technique.

An I/O mirroring technique based on VMWare ESX server has been developed by Mashtizadeh et al. [73]. It works by blocking the read and
write I/O operations on the source server until the VM image is successfully migrated on the destination server. The frequency of big data read and write operations is very high and holding such operations for a small amount of time will create a very lengthy queue. The proposed technique lacks in providing any solution for how to safely block such a large number of read and write operations. However, blocking such large number of read and write operations will result into delayed services that is not desirable for customers and CSPs.

Another technique that uses a combination of on-demand copying and background copying concepts to transfer a disk image rapidly over the WAN having less impact on I/O performance [74]. The proposed solution is implemented as a proxy server using storage I/O protocol (e.g., iSCSI). This solution can be integrated into network attached storage devices available in a data center, hence making it independent of VMM implementations. This technique is indeed promising and can be employed for big data applications, but no testing has been performed from this perspective. Also, other WAN parameters such as latency and bandwidth must be considered and calculations regarding transfer time needs to be performed.

However, all aforementioned approaches induce heavy I/O latencies, especially in the case of I/O intensive workload migration. Nicolae and Cappello [75] have further minimized I/O latencies by reducing data size and network traffic. First one transfers only the modified content excluding the VM’s OS. Second technique is more complex as it involves a hybrid data fetching method in which push and pull methods are used. Cold data are pushed by the source server and the dirty data are pulled by the destination server. The server possessing the transfer-control is only allowed to perform the operation. When we discuss about big data, the time to locate and identify modified contents is very high due to a large volume of data. The process of locating modified contents will consume more CPU cycles. The processing of cold and dirty data is also a CPU intensive task. In addition, the server not possessing the transfer control will have to wait for longer time due to the processing of other server on big data holding transfer-control.

A scheduling technique has been discussed in Ref. [76] that is workload-aware and based on disk block transfer. In the proposed technique, disk blocks are scheduled to be transferred in an optimized way resulting in minimized network traffic and improved I/O performance. The concepts of spatial locality, temporal locality, and popularity characteristics of read, write, and disk block operations are used as key factors to schedule ordering of data blocks. However, the proposed technique consumes more CPU
cycles. Mostly CPU is not considered a major limiting factor in geo-distributed migration, but when big data disk blocks are to be transferred the CPU might become a limiting factor. The reason is that the calculations of spatial locality, temporal locality, and popularity for such a large amount of read, write, and disk block operations will consume a much greater number of CPU Cycles.

In Ref. [77], Jadhav has proposed a scheme to perform a VM disk migration based on the principal of I/O mirroring. An iterative copy approach is used for VM memory state migration. A point-to-point connection between source and destination server is established, and synchronous mirroring technique is used to transfer the VM disk. For big data the number of iterations for copying such a large volume of data will be increased and synchronous mirroring will also take longer time. The chances of errors in the synchronization process will also be increased as link stability over WAN is lesser than in a LAN environment.

From discussion about geo-distributed data center techniques, we can conclude that existing techniques are not big data ready. A comparative analysis of all aforementioned techniques from big data perspective has been provided in Table 5.

4. Summary, insights and open research challenges

A possible future cloud model may consist of SDN and big data that are main ingredients of future Internet [78]. The concept of cloud federation has already been coined, in which CSPs collaborate with each other for resource sharing and load balancing [79]. SDNs can play a pivotal role in the processing of big data due to features such as programmability, control, and efficiency of data transfer [80]. A possible consolidated model of the future cloud has been provided in Fig. 9.

Every CSP in the federation will have a control plane and data plane. Programmability is employed to cope with the volatile network change and reduce cost of the equipment. The control plane will work in a distributed manner connecting to all other control planes. Since control plane is responsible for control traffic that has a low data rate and can be managed easily, higher data rates can be achieved at data plane using traffic flow rules to aggregate and partition traffic flows according to link condition [18]. When a certain migration is triggered, the connections to the affiliated NAS can be transferred temporarily and data shifting process from source NAS to destination can be scheduled in off-peak hours. After the successful
Table 5 Comparison of approaches to improve VM performance overhead for geo-distributed data centers.

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Performance overhead parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Migration time</td>
</tr>
<tr>
<td>Write throttling IP tunneling [35]</td>
<td>✓</td>
</tr>
<tr>
<td>Pre-copy [36]</td>
<td>–</td>
</tr>
<tr>
<td>Correlated VM migration [69]</td>
<td>✓</td>
</tr>
<tr>
<td>Virtual data center migration [70]</td>
<td>✓</td>
</tr>
<tr>
<td>Page cache restoring [71]</td>
<td>–</td>
</tr>
<tr>
<td>De-duplication, data priority transfer [72]</td>
<td>–</td>
</tr>
<tr>
<td>I/O mirroring [73]</td>
<td>✓</td>
</tr>
<tr>
<td>On-demand fetching, background copying [74]</td>
<td>–</td>
</tr>
<tr>
<td>Modified data transfer, data pre-fetching [75]</td>
<td>–</td>
</tr>
<tr>
<td>Data block scheduling on locality, popularity [76]</td>
<td>✓</td>
</tr>
<tr>
<td>I/O mirroring [77]</td>
<td>✓</td>
</tr>
</tbody>
</table>
migration of data, the connections to old data can be terminated. A grace period must be set in order to make sure that the migrated VM and data is successfully transferred to the destination site and work as required. After the expiration of that grace period, the respective VM and data can be removed.

An amalgamated view of the VM performance parameters has been presented in Fig. 10 along with the summary and future directions keeping big data in perspective. It is evident from the overall discussion in this chapter that much work has been done in the area of shared resource management and LAN migration. However, areas such as WAN migration, downtime, and especially big data storage migration are still open research issues.

4.1 VM migration and big data

VM migration techniques serve as a base for managing computing resources, minimizing VM performance overhead, achieving energy efficiency and load balancing in cloud computing [77,81].

Currently, the aforementioned VM migration techniques lack in handling the velocity, variety, and variability of the data. In WAN, the bandwidth requirements for such—data intensive applications are very high posing another challenge for the low bandwidth networks. The network
infrastructure also lacks in accommodating such large sized requests. On a low bandwidth network, aforementioned techniques will perform poorly due to the delayed migration as frequent updates on the source image will generate more network traffic. However, making the VM design less dependent upon applications can solve the problem. For example, for a web server if only PHP and MYSQL is required to run on the server then we can only move the application and its data instead of migrating the whole VM in a standard web server VM image.

4.2 VM storage migration and big data

VM migration techniques manage the performance overhead in a cost effective manner. When discussing about performance overhead management, a VM migration technique emphasis on two methods through which the desired results can be achieved. One is to reduce the size of data to be...

Fig. 10 Consolidated view of the VM performance parameters.
transferred during the VM migration process between source and destination server and second is to reduce the amount of memory requirement or disk dirty rate.

Techniques in Refs. [36,60–63,72] fall in the category of minimizing the size of data transferred or disk dirty rate. What about the storage migration which is a vital part of the migration process? Big data storage drives are difficult to migrate along with VM images generating more data and network traffic. Second, the techniques falling in the category of minimizing the size of data transferred or disk dirty rate achieve their goal by blocking I/O requests to the migrating VM, while in the process of migration (e.g., [72,73]). In case of big data, where the number of write requests is very high, there should be a large enough buffer to temporarily hold the incoming requests. Otherwise, the users will face downtime of services. However, if the option of buffering such larger requests is considered, the load will definitely shift to computing resources such as CPU and memory effecting the collocated VMs and applications running on them. Hence, the design of such an approach that will cover the objective of consuming less computing resources without affecting the collocated VMs is still a challenge. If the need of storage migration can be eliminated in a way that storage can be accessed from anywhere, then the relevant data can easily be assigned to the newly migrated VM. But the solution for assigning data to VM will not work for a data intensive application due to communication latency introduced by the newly assigned route and channel over WAN.

4.3 Effect of VM migration on collocated VMs with big data

The extensive study about VM migration techniques shows that the majority of the techniques are either focused on the performance overhead caused by VM migration or on the effects of migrated VM on source and destination servers (e.g., [50,56]). There are not many techniques that have discussed the effects on VM collocation on a single destination server due to live migration [58]. As the VM is transferred to a new physical server, it fights for the limited physical resources, causing an effect on the collocated VMs hosted on the destination server. However, this is true for both source and destination servers as for time being during migration, additional resources are required on the source side, and after migration, the resources are released providing more room for the collocated VMs on the source server.

Big data storage migration is more challenging as many distributed algorithms work on storage and databases. It becomes hard to establish a global
picture in the presence of distributed algorithms as the frequency of data change and WAN latency is very high. It is very critical to transfer the relevant information, schema, and data with the VM. It is required by the VM performance overhead mitigation techniques to consider both factors causing the overhead, VM performance overhead due to migration of VM and storage, and its effects on collocated VMs on source and destination servers. Establishing such a performance centric approach can be a challenging research problem. However, by improving VM resource prediction techniques for destination server, it is possible to provision required resources in advance, thus avoiding migration effects on collocated VMs at run time.

4.4 Cost effectiveness and big data

In cloud data centers, finding a tradeoff between the monetary cost of hardware solution (i.e., network switches, servers, storage drives, and network links) and VM performance overhead is an appealing research area. With the increase of data and network traffic, it has become inevitable to work with low bandwidth and simple hardware. The management and processing of big data requires complex and speedy hardware. A simple solution could be rendering more hardware for the same number of VMs. It will definitely help in improving performance but will increase the cost which is not a desirable factor for the cloud service providers. In an extreme case, it is possible to create a one-to-one mapping solution, that is one VM to a single physical server, one logical link to single physical connection, one VM storage to a physical storage, but the resulting solution will become highly impractical. However, in this example the resource utilization will be very low and VM performance overhead is either at minimal level or removed completely which is against the interest of the cloud service providers. Hence, this research is very vital to evaluate the tradeoffs between required physical resources, their utilization, and sharing, keeping the performance overhead factor and big data challenges in perspective. A combined effort in all directions is required to produce cost effective solutions using less sophisticated and low computing hardware requiring less bandwidth and network resources.

4.5 Network traffic, heterogeneity and big data

Majority of the network sharing solutions are those that are based on the hose model (e.g., [82–86]). In hose model, dedicated network connections are used to connect all VMs with a non-blocking logical switch. Network
capacity of the central switches is not infinite and hose model only provides simplification for the VM’s networking paradigm. Network congestion cannot be avoided completely whether a small or large data center network with the enormous amount of bandwidth [87] is considered. Network congestion is the main reason due to which hose model is impractical in real world data centers. Hose model also does not provide any means to manage the greater velocity of the data over the network. The question still remains, that how a network model should be designed to facilitate the network sharing solution keeping the heterogeneity (e.g., [86,88]) and big data challenges in perspective.

The majority of the existing VM allocation techniques have targeted homogenous network formation and bandwidth requirement, few solutions have studied the heterogeneous nature of network and bandwidth requirement which further complicates the bandwidth sharing problem. More focus and research is required to develop VM and network sharing solutions targeting the heterogeneous nature of the network and bandwidth requirement on physical as well as logical level. In a cloud data center, the hardware heterogeneity is possible [89], but even within the same type of Amazon EC2 instances, it does not mean that it is also ready for big data. Accommodating the heterogeneity factor along with big data challenges while applying techniques for VM migration and performance overhead mitigation can be challenging. However, if an intelligent control system can be developed using the programmability feature of devices on the fly as offered by SDNs, the adaptability to work with different networks having different data and network requirements can become easy. The adaptability feature will help in balancing resources according to their unique requirements efficiently and the problem of heterogeneity can be resolved to some extent.

Need for computing resources is highly volatile and this fact has also been proven by Google’s production data center analysis [90]. Different kinds of workloads running simultaneously in the data center makes the resource demand prediction more difficult. Another fact is that the majority of the VM performance management solutions work on the principle of accurate resource prediction, which is very difficult when data variety, velocity, and variability is very high. Accurate resource prediction in heterogeneous and volatile workload with less prediction error is very challenging.

4.6 Hybrid of public and private clouds with big data

It is an observation that the public and private clouds are used in combination forming a hybrid model to facilitate the customers. Customers may
jump between public and private clouds as per their need of resources and are charged accordingly [87]. This private-public hybridization is also used to accommodate the needs produced by big data applications. However, customers are not ready for the related challenges of data capture, data storage, data analysis, data visualization, required bandwidth, velocity, variety, and variability. There is a strong need for designing new applications by keeping the hybridization factor in perspective.

If we take the example of “cloud bursting” [91], where synchronization becomes more critical as the service runs in the local data center and its replica runs on the public cloud. Synchronization of big data will be very difficult as it requires synchronization of VM’s state and the snapshot across the network, thus producing a large volume of data to be transferred between private and public cloud replicas. Are the existing solutions ready to support such heavy and inflexible resource demand? These are all open questions for the future research to address keeping the VM performance and big data in perspective, especially across multiple geo-distributed data centers. However, the problem of synchronization can be minimized if we can develop such scheme for the deployment of VMs having a combination of data intensive and non-intensive applications running on different VMs, but on the same physical server. So, if there is a need for more resources by a VM running data intensive application, the VM having less data intensive application can be migrated instead. Hence, local resources can be provided at priority to the VM having data intensive applications and the other VM having less data intensive application can use Cloud Bursting resulting in less data for synchronization.

5. Conclusion

This chapter presents VM performance issues, effecting factors, and overhead of mitigation techniques. Resource parameters and challenges related to different operations and big data have been established to assess overhead mitigation techniques. It has been concluded that existing VM migration techniques are not ready to handle big data, especially the aspects of big data application and storage migration are yet to be incorporated. A possible future cloud model has been discussed that will help in developing new techniques to address big data challenges. The future cloud model consists of SDN and cloud federation to support big data. The programmability feature of SDN provided by control plane can be leveraged to attain customized underlying network configurations suitable for big data. Moreover, the
data plane of SDN helps in providing sufficient data rate due to separation of control decisions, thus making it suitable for big data applications and storage migration.

References


About the authors

Muhammad Ziad Nayyer is serving as Assistant Professor in Department of Computer Science GIFT University, Gujranwala, Pakistan. He is a full time PhD student at Department of Computer Science, COMSATS University Islamabad, Lahore Campus working under the supervision of Prof. Syed Asad Hussain. He is an active member of Communication Networks Research Center (CNRC). He received his MS degree in Computer Science from Government College University (GCU), Lahore, Pakistan in 2011. His area of interests include Cloud Computing, VM Migration, Mobile Cloud Computing, Cloud Federation, Mobile Edge Computing, Fog Computing, and Cloudlet Computing.

Imran Raza is working as an Assistant Professor in the Department of Computer Science, COMSATS University Islamabad, Lahore Campus, since 2003. He is an active member of Communication Networks Research Center (CNRC). He holds BS (CS) and MPhil degrees from Pakistan. He has been associated with TU Ilmenau, Germany as a researcher. His areas of interests include SDN, NFV, Wireless Sensor Networks, MANETS, QoS issue in Networks, and Routing protocols.
Syed Asad Hussain is currently leading communications and networks research at COMSATS University Islamabad, Lahore campus Pakistan. He obtained his master’s degree from Cardiff University, UK. He was funded for his PhD by Nortel Networks, UK at Queen’s University Belfast, UK. He was awarded prestigious endeavour research fellowship for his postdoctorate at the University of Sydney Australia in 2010, where he conducted research on VANETs. He has taught at Queen’s University Belfast, UK; Lahore University of Management Sciences (LUMS); and University of the Punjab. He is supervising PhD students at CUI and split-site PhD students at Lancaster University, UK in the fields of cloud computing and cybersecurity. Professor Hussain is serving as dean of faculty of information sciences and technology since 2015 and has served in the capacity of head of Computer Science Department COMSATS University Islamabad, Lahore campus from August 2008 to August 2017. He regularly reviews IEEE, IET, and ACM journal papers.
CHAPTER THREE

Toward realizing self-protecting healthcare information systems: Design and security challenges

Qian Chen
Department of Electrical and Computer Engineering, The University of Texas at San Antonio, San Antonio, TX, United States

Contents
1. Introduction 115
2. Autonomic computing and self-protecting HISs 120
   2.1 The employment of control theory to self-protection 121
   2.2 Self-protection model and functionality 123
3. Healthcare information system 126
   3.1 Communication standards 127
4. HL7 messaging standard 128
   4.1 HL7 V2.5.1 data frame 129
   4.2 Security challenges in healthcare information systems 131
5. State-of-the-art research for enhancing HIS security 133
   5.1 Intrusion detection systems 134
   5.2 Network forensics analysis (NFA) 137
   5.3 Intrusion response systems 139
6. Conclusion 142

Acknowledgments 142
References 143
About the author 149

Abstract
This book chapter reviews the history of Healthcare Information Systems (HISs), discusses recent cyber security threats affecting HISs, and then introduces the autonomic computing concept and applies the concept to design self-protecting HISs (SPHISs) that can defend themselves against cyber intrusions with little or no human intervention. To realize such SPHISs, we first study security vulnerabilities of the HIS network, communication links and protocols. Based on these vulnerabilities, the component design of a SPHIS is presented. We propose that a SPHIS should contain monitoring systems, early estimation modules, intrusion detection, network forensics analysis devices and intrusion response systems. Finally, existing self-protecting approaches for HIS, enterprise
systems and industrial control systems are demonstrated in detail. This chapter provides an innovated design of an autonomic security management system that could reduce IT professional’s management burdens while enhancing the security posture of their HISs.

**Abbreviations**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC</td>
<td>autonomic computing</td>
</tr>
<tr>
<td>ADT</td>
<td>admission, discharge, and transfer</td>
</tr>
<tr>
<td>ARIMA</td>
<td>autoregressive integrated moving average</td>
</tr>
<tr>
<td>ASMF</td>
<td>autonomic security management framework</td>
</tr>
<tr>
<td>BCH</td>
<td>Boston Children’s Hospital</td>
</tr>
<tr>
<td>CDA</td>
<td>Clinical Document Architecture</td>
</tr>
<tr>
<td>CIS(s)</td>
<td>Clinical Information System(s)</td>
</tr>
<tr>
<td>CPR</td>
<td>computer-based patient record</td>
</tr>
<tr>
<td>DDoS</td>
<td>distributed denial of service</td>
</tr>
<tr>
<td>DHMG</td>
<td>DeKalb Health Medical Group</td>
</tr>
<tr>
<td>DICOM</td>
<td>digital imaging and communications in medicine</td>
</tr>
<tr>
<td>DNS</td>
<td>domain name system</td>
</tr>
<tr>
<td>ED</td>
<td>emergency department</td>
</tr>
<tr>
<td>EHR</td>
<td>electronic health record</td>
</tr>
<tr>
<td>FIS(s)</td>
<td>financial information system(s)</td>
</tr>
<tr>
<td>HIS(s)</td>
<td>healthcare information system(s)</td>
</tr>
<tr>
<td>HIT</td>
<td>health information technology</td>
</tr>
<tr>
<td>HITRUST</td>
<td>Health Information Trust Alliance</td>
</tr>
<tr>
<td>HL7</td>
<td>Health Level Seven</td>
</tr>
<tr>
<td>HTTP</td>
<td>hypertext transfer protocol</td>
</tr>
<tr>
<td>ICMP</td>
<td>Internet control message protocol</td>
</tr>
<tr>
<td>IDS</td>
<td>intrusion detection system</td>
</tr>
<tr>
<td>IMD(s)</td>
<td>implantable medical device(s)</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of things</td>
</tr>
<tr>
<td>IoMT</td>
<td>Internet of medical things</td>
</tr>
<tr>
<td>IPsec</td>
<td>Internet protocol security</td>
</tr>
<tr>
<td>IRS</td>
<td>intrusion response system</td>
</tr>
<tr>
<td>LOINC</td>
<td>logical observation identifiers names and codes</td>
</tr>
<tr>
<td>MDP</td>
<td>Markov decision process</td>
</tr>
<tr>
<td>MSG</td>
<td>message type</td>
</tr>
<tr>
<td>MSH</td>
<td>message header</td>
</tr>
<tr>
<td>NCPDP</td>
<td>National Council for Prescription Drug Programs</td>
</tr>
<tr>
<td>NFA</td>
<td>network forensics analysis</td>
</tr>
<tr>
<td>NIH</td>
<td>National Institute of Health</td>
</tr>
<tr>
<td>NIST</td>
<td>National Institute of Standards and Technology</td>
</tr>
<tr>
<td>NM</td>
<td>Numeric</td>
</tr>
<tr>
<td>OWASP</td>
<td>The Open Web Application Security Project</td>
</tr>
<tr>
<td>PHI</td>
<td>personal health information</td>
</tr>
<tr>
<td>REDR</td>
<td>risk assessment, intrusion estimation, intrusion detection and intrusion response</td>
</tr>
</tbody>
</table>
1. Introduction

“Medical informatics is the application of computer technology to all fields of medicine—medical care, medical teaching and medical research [1].” The history of medical informatics or Healthcare Information Systems (HISs) in the United States can be traced back to the 1950s with the breakthrough of information technology and the rise of computers. The primary stage of the development of HISs was from the 1950s to 1980s. In this period, scientists developed computer software to relieve human burdens from trivial calculation and daily diagnosis.

In the 1960s billing was the center of the HISs. In the same decade, the National Institute of Health in the USA (NIH) began to support medical informatics projects, which promoted the development of HIS. As a result, administrative and financial information systems (FISs) were developed in large hospitals and academic medical centers to support the management functions and general operations of the healthcare organizations [2]. In the 1970s Clinical Information Systems (CIS) containing patient clinical and health-related information was established. Meanwhile automatized data processing and computer-assistant medical decision-making were growing rapidly. Shared systems were still used in this period, and data processing was primarily centralized on mainframe computers.

The second stage of the modern HIS was from 1980 to 2000 with the introduction of artificial intelligence and the development of the Internet.
and microcomputers. In this period, CIS was continuously expanded worldwide, from large hospitals to small ones. FISs and CIS were finally integrated, and distributed data processing became feasible. In 1991 the Institute of Medicine began recommending healthcare organizations to implement Computer-based Patient Record (CPR) for reducing national healthcare costs while enhancing the care of patients. By the end of 2000, the adoption rate of CPR was much lower than the expectation. Only 17% of 329 family practice residency programs were using a CPR at that time [3].

The third stage of HIS was from 2000 to present when electronic health record (EHR) and e-Health systems were widely adopted by healthcare organizations. According to a recent report from the Office of the National Coordinator for health information technology, by the end of 2015, 96% of the US hospitals possessed a certified EHR technology that met the technological capability, functionality, and security requirements required by the Department of Health and Human Services [4]. From the report, one can observe that the number of hospitals implementing at least one basic EHR significantly increased since 2008, from 9.4% to 83.8%. This adoption rate includes small hospitals in the rural areas. Note that more than 50% of the increment occurred in the 2011–2015 period.

Almost in the same period, the Internet of things (IoT) applications changed the traditional healthcare industry. Today patients can remotely access their medical data by the support of cloud computing technology. The development of mobile devices and healthcare wearables allows both physicians and patients to monitor patient’s health in real time for improved health outcomes. Additionally, the real-time patient monitoring and advanced homecare medical devices reduce the cost of unnecessary physician visits and hospital readmissions [5].

Although the idea of HIS was brought forward in the 1950s, the general utilization of information technology in healthcare sectors has not seen a marked increase until recent 5 years. As soon as we began to enjoy the benefits of the HIS and the Internet of medical things (IoMT) for better quality of healthcare with a low maintenance cost, the HIS has quickly become the top target of cyber criminals. According to the 2016 IBM X-Force survey of assessing and examining the goings-on in the world of cyber security and cyber threats, healthcare was the most frequently attacked industry [6]. Over 112 million health records were compromised in 2015 [7]. Some of highly publicized HIS cyber security incidents include:

1. Phishing: intruders masquerade themselves as trustworthy entities to obtain sensitive information by deceiving victims who are inside of
healthcare IT networks. The phishing attack is one of social engineering attacks. As an example, victims receive email messages appearing to come from legitimate enterprises but actually from attackers. Victims follow attacker’s commands such as downloading malware or granting privilege to the intruders, and finally HISs are compromised and scammers gain access to the database systems and monitor specific actions [8]. Although users are very familiar with the phishing attack, it is still successful and destructive. It has already become a great risk to the healthcare industry [9]. This is because phishing attacks are cheap, easy, and flexible to launch. Moreover non-technical people such as healthcare practitioners and senior patients who are not well trained have become a pool of potential victims.

Recently the number of real-day phishing attacks that successfully compromise HISs have increased significantly. The largest healthcare cyber attack of 2015, Anthem Breach, which affected 78.8 million people, began with phishing emails sent to the employees working at Anthem INC. [10]. Phishing attacks also affected 220,000 patient information records of DeKalb Health Medical Group (DHMG), which was the eighth largest cyber attack in healthcare of 2015 [9, 11]. The ransomware attacks introduced later are one of the consequences of the phishing attacks, through which intruders gained access of the DHMG HIS, and resided in the system, monitored and obtained the organization’s activities from November 2013 to January 2015. Moreover, intruders set up a fraudulent donation webpage, which was similar to the DHMG’s charity donation page and sent phishing emails to ransomware attack victims for financial rewarding [12].

In recent 2 years several severe CEO phishing attacks compromised HISs. For instance, 11,000 W-2s information from healthcare workers of Pennsylvania Main Line Health was compromised via the attack in February 2016 [13]. In the same month, a similar cyber attack compromised St. Joseph’s Healthcare System and 5000 employee’s earning data was stolen [14]. Intruders generally masqueraded themselves as CEOs of their target organizations and sent spoof emails to the employees from a seemingly legitimate source, and eventually gained the access to patient and employee’s information from the compromised HISs.

2. Malware: Computer malware is four times more likely to be seen in healthcare organizations than other industries [15]. Following the initial attack vector such as social engineering attacks, healthcare employees may download and spread malware into their HISs. According to a
study in 2015 conducted by the Health Information Trust Alliance (HITRUST), 52% of 30 mid-sized US hospitals were infected with malware, and the most common type was ransomware [16].

Ransomware attacks can encrypt all files of the compromised information systems to deny legitimate access to the data. One of the most influenced HIS ransomware attacks happened in February 2016. The attackers encrypted Hollywood Presbyterian Medical Center hospital’s data and seized control of its computer systems. The ransomware attacks significantly disturbed the hospital’s normal operations and paralyzed its EHR system. The hospital ultimately paid $17,000 to the hacker for restoring the functions of their computer systems [17]. Following the first ransomware attack, five similar attacks were found in the next several months. The same attack vector was used to gain accesses to information systems of hospitals and healthcare organizations in CA, KY, and Washington DC.

Incidents caused by ransomware attacks may even lead to sensitive personal and treatment data breaches. On August 24 and 25, 2016, Man Alive Inc. Lane Treatment Center was attacked by the ransomware attack that gained unauthorized access to their EHR systems, stole patients’ mental health and substance abuse records, and finally sold the records on a dark web [18]. This attack highlights new challenges that the healthcare organizations can face in securing EHR containing extra-sensitive patient records.

3. SQL Injection: Attackers exploit vulnerabilities of the traditional SQL database applications used in HISs to bypass the authentication process. Intruders therefore access unauthorized Personal Health Information (PHI) that lead to sensitive data breaches. In May 2016, anonymous hackers attacked 33 Turkish hospitals databases via SQL Injection attacks and leaked more than 10 millions Turkish medical healthcare records [19, 20]. It is very likely that the same attack vector was adopted by another team of hackers who compromised personally identifiable information of 1400 employees working at York Hospital in Maine in early 2016 [21, 22].

4. Distributed Denial of Service (DDoS): This is the most common hackvisit attack overloading healthcare servers with adverse traffic to shut down EHR and email systems; thus preventing legitimate requests accessing HIS resources (e.g., critical patient information) [23, 24].

Anonymous attackers have conducted various DDoS attacks to enterprise and healthcare information systems against politicians, companies,
and governments [23]. Boston Children’s Hospital (BCH) was the first healthcare organization targeted by a DDoS attack in 2014 “in response to the diagnosis and treatment of a 15-year-old girl who had been removed from her parent’s care by the Commonwealth of Massachusetts [23].” The DDoS attacks began with a threat and has involved three major strikes for several months [25]. The hospital immediately reacted to and mitigated the attacks with the help of third-party security companies. Even though the DDoS attacks did not significantly damage BCH HISs, their experience shows that every healthcare entity is the potential target and victim of DDoS attacks.

Healthcare and public health is one of 16 critical infrastructure sectors [26]. Unlike other computing systems, HISs are lagging in terms of proper cyber security defense and investment [27]. Medical information, however, is valuable for hackers and this makes HISs popular targets for cyber attacks. One recent survey shows that 340% more information security incidents occurred in the healthcare sector than other industries [28]. According to 2015 KPMG Healthcare Cyber Security Survey [29], only 53% of healthcare providers and 66% of health insurers out of 223 US-based healthcare organizations said that they are prepared to defend against an attack. 81% of healthcare organizations have been compromised by cyber attacks in the past 2 years.

Current healthcare organizations continuously rely on security solutions such as antivirus, firewalls and data encryption to secure their IT environment. However, HIS security is still largely at risk from sophisticated Brute Force, Phishing, Malware, SQL Injection, and DDoS attacks [30]. The biggest barriers of healthcare organizations to mitigate cyber security events are lack of financial resources and appropriate cyber security professionals. The large amount of emerging threats and the complex nature of current network infrastructure made the HIS protection even harder [30]. Therefore, there is a great urgency to develop more advanced and cost-efficient security solutions to reduce the human burden of managing and mitigating HIS threats and vulnerabilities.

Autonomic Computing (AC) technology was inspired by the autonomic nervous system that regulates and maintains homeostasis from the subconscious efforts of the brain. In an analogous way, AC technology allows computing systems and applications to manage themselves with minimum human intervention. The self-protection capacity has been applied to securing enterprise computing systems and Supervisory Control and Data Acquisition (SCADA) systems from both internal and external known or zero-day threats.
attacks with little or no human intervention [31–36]. A similar approach can be developed to supplement the system administrator’s responsibility to anticipate and defend the HIS as a whole from malicious activities.

2. Autonomic computing and self-protecting HISs

The self-protection aspect of autonomic computing can be used to secure and safeguard HIS communications, properly grant authorization privileges, and evaluate performance of security mechanisms toward achieving greater resiliency. By continuously monitoring measures of performance and security, autonomic systems can ascertain differences associated with normal vs abnormal system behavior. With the usage of mathematical techniques for intrusion detection and response evaluation algorithms, the autonomic approach can efficiently and automatically respond to a wide range of cyber attacks. Since human capacities cannot possibly protect large-scale complex HISs from increasingly sophisticated cyber adversaries combined with high volumes of network traffic, the application of autonomic computing can supplement or replace human intervention needed for the management of cyber security posture and situational awareness.

Kephart [37] and Parashar and Hariri [38] summarize that an autonomic computing system must possess eight key properties:

1. Self-awareness: the computing system needs to “know itself” and is aware of its state and behaviors.
2. Self-configuration: the computing system should configure and reconfigure itself under various circumstances.
3. Self-optimization: the computing system should continuously optimize itself to enhance the operation.
4. Self-healing: the computing system should recover itself from failures.
5. Self-protection: the computing system should proactively detect and protect itself from cyber attacks.
6. Context awareness: the computing system should know its environment and act accordingly.
7. Open environment: the computing system should function within an open environment.
8. Estimated resource allocation: the computing system should optimize and anticipate its needed resources.

Therefore, to establish an autonomic computing system, one must employ control theory that provides finite methodologies for diagnosing and repairing system problems. This can be achieved with the utilization of feedback loops,
autonomic managers, and means to communicate with surrounding environment. After analyzing this information to determine the desired (baseline) values of the observables (reference inputs), autonomic managers construct and execute plans to regulate systems based on current system states [39, 40]. Fig. 1 [39] illustrates the autonomic computing model, which is the combination of autonomic elements and feedback control theory.

In large-scale distributed environments, autonomic computing is a promising technique to enable systems to self-manage without requiring human commands. It not only successfully configures, optimizes, and heals managed elements, but also protects computer systems from cyber attacks. In the following sections, essential components, theories, and techniques employed to realize self-protecting HISs are presented. First, the control theory and its application to self-protecting systems are discussed. After that the functionality of self-protecting HISs is introduced in detail. Finally, state-of-the-art self-protecting HISs are surveyed.

2.1 The employment of control theory to self-protection

Self-protecting systems proactively anticipate cyber assaults relying on early reports generated by sensors. These systems also take steps to eliminate or mitigate cyber attacks with little or no human intervention. They adopt
the feedback control theory to monitor managed elements, analyze system performance, and select and execute appropriate plans continuously to enhance security, reduce the operator load from system management, and consequently reduce human error and cut operational costs.

The block diagram of a self-protecting system adopting the feedback control loop shown in Fig. 2 is taken from a paper by Diao et al. [40]. Similar to Diao’s research, self-protecting HISs use feedback control theory to regulate user connections in the context of a constantly changing operating environment. The same type of feedback control is used to describe how self-protecting functionality is realized.

The essential elements in Fig. 2 for implementing a self-protecting HIS are described as follows:

- **Target System**: This refers to the victim system that is attacked by adversaries. For example, the victim can be a multilayered web application environment of the HIS, which contains web servers, application servers, and database systems; or possibly only a single database server providing patient information.
- **Reference Input**: Reference inputs are the desired values from target system outputs. In reference to security aspects, the desired system outputs are: (1) providing “normal” services to legitimate clients and (2) eliminating and protecting systems from all cyber assaults.
- **Control Input**: Control inputs are effective countermeasures or responses that are able to eliminate or mitigate cyber attacks thereby improving the security of HISs within their specific operating environments.
- **Control Error**: Control errors result from large differences between the Reference Input baseline and current observations. The differences could be Euclidean distances between measured system states and the “normal region” of a secured system. The differences also provide the basis for

![Feedback block diagram](image-url)

*Fig. 2 Feedback block diagram [40].*
determining the likelihood that the system performance is within the “normal” operating range.

- **Controller:** The controller evaluates recommended responses or policies, and initiates the most appropriate responses needed to recover the compromised HISs considering control environment inputs.
- **Disturbance Input:** Any cyber attacks undermining confidentiality, integrity, and availability of HISs are represented by disturbance inputs. Disturbance inputs may include malicious requests, unauthorized accesses, spoofing, DDoS attacks, and eavesdropping (i.e., man-in-the-middle).
- **Noise Input:** This refers to missing data or noisy data collected by sensors.
- **Transducer:** It transduces system security states in real time to enable the security states of the *Target System* to be compared with the *Reference Inputs* (i.e., “normal region”) to ascertain the Control Errors.
- **Output/Observation:** The output of the controller is the observation of the *Target System* security states, which is composed of real-time values such as system, network, and security features of the servers or power consumption of medical devices.

### 2.2 Self-protection model and functionality

To better understand how a self-protecting HIS is architected, we developed a comprehensive autonomic security management framework that contains Risk assessment, intrusion Estimation, intrusion Detection and intrusion Response (REDR) modules. Fig. 3 shows the REDR framework that gives a model for adopting control theory to assess vulnerabilities and to baseline the normal region of the HIS behavior for a specific operational environment. The self-protecting HIS, as the first-line of defense, anticipates

![Fig. 3 REDR an autonomic security management model.](image)
upcoming anomalies, sends early warnings to the controller, and protects the system by signaling the controller to initiate an active response to selected threats. Sophisticated attacks that evade the first-line of defense due to incorrect estimations of future system security states may be thwarted by an intrusion detection system (IDS). Live network forensics analysis is used to learn the causes and impacts of unknown attacks. Novel signatures [33, 41, 42] are utilized to update detection algorithms and the active response library. As a result, similar attacks can be removed in the future by the first-line of defense. The functionality of each module is discussed as follows:

- **Risk assessment**: The first step to establishing a self-protecting HIS is to assess its security risks offline. Risk assessment helps an organization to determine the impact and likelihood that a given threat will and can successfully exploit a particular vulnerability. HITRUST Risk Analysis Guide introduces a risk management framework in four steps [43]. In our REDR model, the risk assessment module is divided into five steps (as shown in Fig. 4). We added the component of ‘characterize HIS’ before the four steps guided by HITRUST. This component analyzes and characterizes the HIS performance with the help of expert knowledge offline, followed by the establishment of the “normal region,” which represents secure system performance using normal datasets of relevant features. These features can be parameters such as CPU time, memory utilization, I/O read/write rate, packet received/sent rate, TCP connection performance, communication protocol message header and payload that impact HIS’ system, network, and security resources.

System administrators can identify risks to their HISs in step two. Outputs from this module indicate the degree to which the HIS is vulnerable. This step also must determine protection requirements to minimize high-risk areas of HISs. The third and fourth steps are

![Fig. 4 Risk assessment steps and modules.](image-url)
to select, implement and manage comprehensive and/or baseline security controls based on the risk assessment result, which in turn prioritizes the active response to abnormal unwanted or unauthorized activities. Examples of these controls include firewalls, antivirus software, access control lists, security policies, and honeypots.

The last step is to assess the controls selected and implemented in steps three and four, and report the efficiency of the selected controls for mitigating risks to business associates and system administrators. If these controls cannot prevent HISs against known risks, more efficient controls must be reselected and implemented.

Features monitored by the intrusion estimation module are the same parameters used to determine the “normal region.” Real-time observations are collected by the system monitors.

- **Intrusion estimation:** System sensors continuously monitor selected features. Estimation of the system security state uses historical observation to project future feature values using statistical methods combined with system models. Upcoming attacks are therefore anticipated by comparing the “normal region” with the estimated outputs of the system model using forecasting methods (e.g., time series forecasting methods such as Kalman filter or the autoregressive integrated moving average (ARIMA) method [33, 44]). Early warnings, therefore, are sent to the controller, and appropriate control mechanisms are executed to neutralize attacks (on-the-fly).

- **Intrusion detection:** The IDS is a data mining tool that enables real-time event analysis. This module provides accountability for intrusive activities thereby detecting anomalous system behavior. To this end, three types of IDSs (i.e., host-based, network-based, and hybrid IDSs) must be developed using signature and anomaly techniques. The anomaly detection technique comparing real-time system performance with the normal system model detects known and unknown attacks. The signature detection technique identifies known attacks relying on matching observations to known misuse patterns. To identify zero-day attacks (i.e., attacks that exploit previously unknown vulnerabilities using perhaps unknown tactics and methods) the signature database must be upgraded frequently. Using appropriate detection techniques can identify both known and unknown attacks with low false alarm rates. In Fig. 3, the intrusion detection module has the ability to analyze evolving zero-day attacks. This functionality is realized using live forensics analysis tools to learn unknown attack signatures. These signatures are adopted by online detection algorithms and defense mechanisms.
As a result, the previously unknown attacks can be autonomously incorporated into the corpus of known attacks used subsequently by the first-line of defense.

- **Intrusion response**: The utilization of intrusion response systems (IRSs) enables the self-protecting HIS to mitigate attack impacts and regulate system behavior back to normal. IRSs can be divided into static and dynamic types. A dynamic IRS is commonly used to enable an ASMF system. This is because it allows the execution of lower ranking responses for the protection of the compromised system if the so-called optimal responses are not sufficient to mitigate attack impacts. If lower ranking responses are implemented and successfully regulate system performance, the dynamic IRS will escalate rankings of these responses by modifying the initial values of the assessed criteria. Examples of relevant criteria include response effects (e.g., how fast can responses regulate system behavior back to normal), operational costs, and impact on system’s services.

Feedback control theory is the primary means used to realize self-protecting HISs. Therefore, after the self-protecting system realizes all of the steps mentioned above, sensors will still continuously monitor selected features to identify whether system performance does actually return to normal running behavior. When abnormal behavior reoccurs, the system will repeat the self-protection processes mentioned here until the system reaches the baseline normal operational range of states.

### 3. Healthcare Information System

Before we introduce how to realize a self-protecting HIS, let us first understand the computer network, components and communication standards of a typical Healthcare Information System (HIS). The HIS normally contains FISs and CISs. Similar to an enterprise system, a HIS adopts a three-tier client–server architecture that contains a web tier, an application tier, and a data tier for retrieving and storing healthcare related information. As shown in Fig. 5, the web tier, also known as the presentation tier, gives patients and health providers access to a server-based application visible from the graphical user interface via desktops, tablets or mobile phones. The application (logic) tier contains mobile and web applications, which manages business logic, processes commands, and transforms data. The data tier collects operational and business data from CISs and FISs and stores the data in databases or repositories.
3.1 Communication standards

Communication standards for clinical, image, prescription and administrative data transmission are different. Fig. 5 presents a clinical system, radiology information system (RIS), pharmacy system, billing system, registration system, and reporting system. Relevant data of these FIS and CIS are collected and stored in the clinical and financial databases. To communicate with the data tier, each information system must follow individual protocols. HL7, DICOM, NCPDP, X12N, ADT, and LONIC are communication standards widely used for healthcare CIS and FIS. These standards are introduced as follows:

- Health Level Seven (HL7) is an interoperable standard allowing disparate healthcare applications to exchange, integrate, share and retrieve key sets of clinical and administrative data [46]. Therefore, data transmission between the clinical system, billing system, reporting system, and the databases should follow the HL7 standard. More details of the HL7 messaging standard is presented in Section 4.

- Digital imaging and communications in medicine (DICOM) regulates the standards for exchanging digital radiology and cardiology images between RIS and other CIS [47].
• The National Council for Prescription Drug Programs (NCPDP) creates standards for electronic communication of prescription, billing, and other pharmacy material. The NCPDP SCRIPT Standard “supports messages regarding new prescriptions, prescription changes, refill requests, prescription fill status notification, prescription cancellation, medication history, and transactions for long-term care environment.” The NCPDP Telecommunication Standard “supports the electronic communication of claims and other transactions between pharmacy providers, insurance carriers, third party administrators, and other responsible parties [48].”

• X12N is the standard that enables the electronic exchange of healthcare insurance, claims processing data and billing information. Similar to the HL7 standard, X12N can be used for retrieving data from billing systems and reporting systems.

• Admission, Discharge, and Transfer (ADT) messages carry patient information such as patient ID, medical record number, age, name and contact information. ADT messages can also provide “trigger events such as patient admit, discharge, transfer, registration, etc [49].” Therefore, in Fig. 5, the registration system transfers patient information via ADT messages.

• Logical observation identifiers names and codes (LOINC) is the standard for identifying medical laboratory observations. Clinical and laboratory LOINC standards enable the electronic exchange and gathering of clinical results such as laboratory tests and research. After that, the clinical results are recorded into clinical reports and documents.

4. HL7 messaging standard

This section introduces one of the most widely used HIS communication standards, Health Level Seven (HL7), and its security risks. HL7 was developed by a nonprofit organization providing a comprehensive framework and developing standards for the communication of electronic health information [50].

HL7 was founded in 1987. Currently, it has more than 2500 members representing 90% of HIS vendors from 55 countries. The recent report [50] shows that 95% of US healthcare organizations are using the HL7 Version 2.x (V2) messaging standard for exchanging health information in the clinical domain. HL7 V2 is also considered as one of the most widely implemented standards for healthcare in the world. From HL7 V2.5 to the latest HL7 V2.9
released in 2016, these HL7 messaging standards are foundational and mandated by the US Meaningful Use legislation, which means that adopting HL7 standards can “best inform clinical decision at the point of care [51].”

The 2016 Interoperability Standards Advisory (2016 Advisory) recently published a report of the best available standards and implementation specifications [52] based on two rounds of public comment and recommendations from the health information technology (HIT) Standards Committee. The report evaluates HIT standards, and categorized the evaluations in three sections, and they are: best available vocabulary/code set/terminology standards and implementation specifications; best available content/structure standards and implementation specifications and best available standards implementation specifications for services.

HL7 V2.5.1 was named the best available standard for admission, discharge, and transfer; interoperability need, receiving electronic laboratory test results, ordering labs for a patient, etc. In addition, HL7 V2.5.1 is highly adopted in health care within the United States. Therefore, the security of communication under the HL7 V2.5.1 has become a significant challenge. In the following sections, we first introduce the structure of HL7 V2.5.1 data frames in Section 4.1 to have a better understanding of the security risks of the HL7 V2.5.1 message standard, and then we create a message sent from a client (such as from a doctor’s office) to the servers to illustrate the meaning of the MSH’s (message header) fields. We also discuss the security challenges in HIS with the HL7 MSH message as an example (shown in Section 4.2).

### 4.1 HL7 V2.5.1 data frame

HL7 V2 message standards contain a segment and some fields. The content of fields is primitive data types such as Numeric (NM) for sequence numbers and String Type (ST) for field separators. The second type of the field is the composite data type, which contains one or more components. For example, message type (MSG) is a composite data type containing three ID types such as message code, trigger event, and message structure IDs. The fields of the HL7 message can be repeated.

An example of the HL7 message is shown as follows:

HL7 V2.5.1 contains 144 segments covering all clinical and financial activities of the patients, doctors (hospitals), pharmacies, and insurance companies [53]. The segment of the example in Fig. 6 is “Message Header” (MSH). The MSH segment is shown in every HL7 message type and defines source, purpose, destination and specifics of the syntax of the messages.
Besides the segment, the HL7 V2.5.1 MSH messages contain up to 21 fields counting from MSH.1 to MSH.21. In Fig. 6, we wrote MSH.2 as M2 for short to save space. Seven of these fields are required, and they are described as follows:

1. MSH.1 defines the field separator (|). In Fig. 6, we did not name M1 since MSH.1 is the field separator “|”, and it is also used as the separator to separate MSH.1 and MSH.2.

2. MSH.2 defines other separator characters for the message in the order of: a component separator (^), a field repeat separator(~), an escape character separator (\) and a subcomponent separator(\).

3. MSH.7 defines the date or time of the message. In Fig. 6, the time for sending the MSH message is May 2, 2016 at 10:30 am.

4. MSH.9 defines the type of message and trigger events. In the sending application, “VXU” is the message code that shows that the message contains unsolicited vaccination record update information, and the trigger event number, “V04,” matches the message code. “1” is the message structure. These three components are separated by component separators (^).

5. MSH.10 defines the message control ID, which is a number or unique identifier for the message. As shown in Fig. 6, we created ‘SSUCTL-1’ as the control ID, which is a string type.

6. MSH.11 defines the processing ID. In Fig. 6, the processing ID is “P”.

7. MSH.12 defines the HL7 version ID. Version IDs are used for message transmission and help the server side to interpret the messages with an appropriate HL7 version. Since we selected the HL7 V2.5.1 MSH message as an example, the version ID shown in Fig. 6 is 2.5.1.

The example MSH message shown in Fig. 6 also contains another four optional fields from MSH.3 to MSH.6. The description of these fields are as follows:

- MSH.3 defines the sending application. Fig. 6 shows that this message sends from Savannah State University Health Center (SSU HC).
- MSH.4 defines the sending facility. Fig. 6 shows that the sending facility is Lab–1, which can be an immunization and testing lab.
• MSH.5 defines the receiving application. In Fig. 6, the receiving application is Savannah State University Open Enrollment System (SSU OE).
• MSH.6 defines the receiving facility. Fig. 6 shows that the SSU OE system is located in Room-1 (RM1).

More details of another ten optional fields of MSH messages and other 143 segments can be found in this website [53].

4.2 Security challenges in healthcare information systems
Most healthcare organizations rely highly on commercial security products. However, due to practical limitations and unique drawbacks of these security solutions, HISs are far from attack-proof. Scanning attacks, injection attacks, broken authentication and session attacks and DDoS attacks are most dangerous HIS threats. In this section, we first introduce the features and adverse impacts of these attacks, and then we use the HL7 MSH message shown in Fig. 6 to illustrate how these attacks will be simulated, detected, and mitigated.

1. Scanning attacks: Adversaries scan devices in HIS to gather network information of these devices before launching sophisticated attacks to undermine the HIS security. Commonly used scanning techniques to gather computer network information include IP address scanning, port scanning, and version scanning.

   Besides the normal IP address and port scanning techniques (e.g., address resolution protocol and TCP SYN scans) to gather the server IP addresses and the opening ports for the HL7 message transmission, adversaries can also carry out segment scanning attacks to learn message, personal identifiers, order numbers, or patient visit information of specific HL7 messages. Malicious hackers can also obtain the HL7 versions by eavesdropping the version ID field of the MSH messages.

   To protect the HIS against scanning attacks, it is better to close ports that are not commonly used. In addition, relative intrusion detection and protection rules must be updated to drop malicious requests sent to the ports that HL7 servers are listening on.

2. Spoofing attacks: Spoofing attacks happen when adversaries pretend to be legitimate users. Masquerading and impersonation are two types of spoofing attacks. Masquerading is a passive attack. Attackers first exfiltrate legitimate account credentials and then log in to the HIS as legitimate users. Impersonation is also known as replay attacks. This spoofing attack is more active than masquerading. Attackers...
capture authentication traffic and replay that traffic to gain access to the HIS. Once adversaries control the compromised HIS, they can exfiltrate confidential data, exhaust system resources, or spread malware to compromise other networks.

To detect and protect the HIS against spoofing attacks, server sides must frequently change the authentication credentials and enable duplicate detection techniques.

3. Injection attacks: In the data tier, adversaries can exploit the vulnerabilities of structured query language (SQL), JavaScript, and computer programs to trick the interpreter by inserting untrusted data. As a consequence, adversaries may obtain access to the database, attack web users, and propagate computer worms. On the other hand, in the HIS network, adversaries may inject malicious segments, commands, or responses to reduce the security of HIS.

To this end, the HIS messages must be encrypted, and the senders must pass the authentication. The HIS can also avoid SQL queries from users or switch to nonSQL databases.

4. Broken authentication and session management: Attackers exploit vulnerabilities in authentication mechanisms to assume legitimate users’ identities. A successful attack may compromise all accounts. As a result, adversaries can do whatever legitimate users could do. The brute force attack is an example. It takes advantage of weak passwords and small encryption keys. Such attacks send guessed values of account usernames, passwords to the server. Attackers may repeat sending their guesses until they successfully access system resources from compromised accounts.

The EHR servers must strictly limit the account access attempts to avoid brute force attacks. Meanwhile, the IP addresses of the malicious users must be added to the detection rules for filtering malicious packets in the future.

5. Distributed denial of service attacks: DDoS attacks are intended to make HIS resources unavailable using various methods and tactics to exhaust system and network resources. One example is flooding-based DDoS attacks that send a tremendous number of packets to swap web servers. In this way, bogus requests block legitimate requests by overwhelming system’s resources (e.g., CPU, memory, bandwidth).

The best way to protect the HIS from DDoS attacks is to hide the systems (e.g., IP addresses) from users. The flooding messages can be redirected to honeypots. The information of attackers such as IP addresses, protocol types, and source port information must be updated for the detection and protection rules.
5. State-of-the-art research for enhancing HIS security

Sections 3 and 4 introduce HIS networks, communication standards and the potential security threats. In this section we present state-of-the-art research on enhancing HIS security. The HIS can be categorized as an enterprise system and hence, the self-protecting feature in HISs is still in the initial stage. Therefore, besides surveying research on enhancing HIS security posture, we also reviewed existing technology to protect information systems in relevant industries (i.e., distributed enterprise systems and supervisory control and data acquisition (SCADA) systems).

Self-protecting systems know their environment and proactively enhance network and information security. As described in Fig. 3, before a self-protecting system can be useful, the system boundary (e.g., routers, firewalls, web servers, application servers, databases) must first be determined so that those security controls can be baselined during the risk assessment phase. Abnormal system behavior is then a quantifiable deviation from the baseline. Anticipated abnormal system behavior is the early stage of a departure from the baseline. Self-protecting HISs (SPHISs) compare each new request to the anticipated behavior, followed by the execution of an appropriate control. One example of self-protection in a distributed system is introduced by Claudel et al. [54]. Data from the sensors and actuators of the proposed system are utilized to identify abnormal (malicious) requests and isolate compromised nodes automatically. Chen et al. [32, 33] adopted the idea of autonomic computing and created self-protecting systems that can estimate known attacks before they impact devices in the network. If an attack bypasses an intrusion detection system (IDS), a second line of defense provides patterns of known attacks to an intrusion response system (IRS). A controller in the IRS evaluates candidate protection mechanisms considering predefined criteria and then selects the optimal protection mechanisms to regulate system behavior back to normal. Unknown attack patterns were investigated by an online-learning module and were sent to the IRS similar to the known attack scenario. As a result, the self-protecting systems were validated to protect SCADA systems against both known and unknown attacks in near real time with little or no human intervention.

Compared with various self-protecting systems implemented in enterprise and industrial control systems, research on applying autonomic computing to secure HISs is not very mature. Hsieh [55] proposed a framework integrating access control policies, digital signatures and encrypted data into the Clinical Document Architecture (CDA) to realize self-protecting...
e-Health CDA documents. This work enhanced patient data integrity and privacy, however, the self-protecting functionality is very limited. The system cannot react to breaches or learn novel attack patterns. Another group of researchers extended the autonomic computing concept and created a self-managed healthcare Emergency Department (ED) system. This autonomic computing system reduces the dependency on human management and improves system operations and decreases service cost [56]. Lupu et al. proposed a self-managed cell as an architectural pattern for a ubiquitous computing environment to monitor patient’s health in hospitals or at home [57]. However, neither of these two designs include the self-protection feature; therefore, the self-managed ED systems and the healthcare monitoring system are very likely to be attacked.

Although none of current self-protecting approaches have been fully realized as a complete security design for HISs based on the autonomic computing concept developed by Kephat et al. [39], the techniques and approaches used in other computing systems can help us realize a SPHIS. For example, we can adopt the perfect low-cost monitoring system developed by Salih and Zang [58] to contentiously collect patient health data or monitor electrocardiogram data obtained in real time using the Pandey et al.’s design [59]. We can also study from multidisciplinary approaches presented in [60] to achieve an efficient and trustworthy e-Health monitoring system.

In Sections 5.1–5.3, we review the intrusion detection, network forensics analysis and intrusion response technologies that can be applied to secure HISs and eventually realize a self-protecting HIS.

5.1 Intrusion detection systems

An intrusion detection system (IDS) is a data mining tool used to identify cyber attacks. Besides quickly identifying attacks, it has many other benefits such as enabling the collection of intrusion information, recording malicious events, generating reports, and alerting system administrators by raising an alarm.

Recent literature shows that there are several challenges to overcome for today’s IDSs to achieve their goals. The first challenge is the slow speed at which IDSs detect anomalies, especially for signature-based IDSs that match observations to misuse patterns. The second challenge is that most prevalent IDSs cannot detect or adapt automatically to novel zero-day attacks. Most existing IDSs deployed in HISs use a signature-based technique. Known attacks can be easily classified based on attack patterns but not zero-days. On the other hand, by adopting the “anomaly detection technique,” IDSs
can detect both known and unknown attacks. However, without proper classification, deploying accurate counter measures is impossible. The third challenge of current IDS research is the high rate of false alarms. An IDS is an important component in a comprehensive security design, and many security solutions cannot be deployed without an accurate attack detection capability. Moreover, inaccurate detection will adversely affect response selection by the IRS. The obvious result will provide attackers with greater opportunity to compromise the whole system. The following paragraphs overview recent research addressing these three challenges.

The bottleneck of detection speed often plagues signature-based IDSs. This type of IDS must match traffic patterns to a list of rules. The detection speed is influenced by the length of the shortest pattern of rules and the memory access latency [61]. To accelerate the detection speed, recent research has deployed a variety of fast pattern matching algorithms such as the modified Wu–Manber algorithm [62], the set-wise Boyer–Moore algorithm [63], and the FNP algorithm introduced in [61]. Tan et al. [64] extracted partial sample strings from patterns of attacks to enhance the implementation speed.

Using modern graphics processors is another efficient method to address the first challenge. Gnort is an example that yields a detection speedup from two to ten times [65]. Researchers demonstrated that a vulnerability specification and its optimized architecture called VESPA [66] can reduce the time to detection. VESPA uses design principles like fast primitive matching, explicit state management, and parsing of relevant message parts.

The second challenge of today’s IDSs is the inability to detect zero-day attacks or classify known attacks. Bolzoni et al. [67] demonstrated an IDS called Panacea, which integrates anomaly based with signature-based IDSs using machine learning methodologies to address these challenges. The adoption of support vector machine and RIPPER rule learner techniques allows Panacea to classify known and unknown attacks. Another way to enhance the detection rate is to use a behavior rule specification-based IDS for detecting unknown attacks against medical devices embedded in a medical cyber physical system [68]. Boggs et al. [69] examined user-defined content from HTTP requests to detect zero-day attacks across multiple web servers. The abnormal requests were identified by local detection sensors. This approach is very efficient at detecting previously unseen attacks with a false positive rate of only 0.03%. However, this method only examines HTTP attacks and cannot detect attacks that exploit vulnerabilities of other network protocols (e.g., HL7, TCP, ICMP, or DNS).
Amann et al. [70] presented an input framework that integrates external information into the IDS decision process. External information is the third-party intelligence that includes known botnet servers, malware registries, and blacklists. Frequently updating external intelligence does not cause a delay in detecting anomalies; however, this framework may bring additional security concerns from attackers who game the system. Artificial immune systems inspired by their biological counterpart have been applied to scan network traffic and identify zero-day attacks. Danforth [71] developed a web classifying immune system (WCIS) employing an artificial immune system, which achieved a high rate of accuracy for identifying and classifying SQL injection attacks, XSS attacks, and buffer overflow attacks.

The third challenge of IDSs is how to improve the accuracy of the detection rate (i.e., reduce false positive/negative). In a large-scale distributed system, the choice and placement of detectors influence the accuracy of the overall detection function. Modelo-Howard et al. [72] used a Bayesian network model to evaluate the effects of detector configuration and placement against the accuracy of the detection rate and the likelihood of an attack goal being achieved. Selecting the correct features to monitor is another key to improve detection rates. Chu et al. [73] built a user behavior model adopting statistical methods based on device access logs to catch anomalies and protect the networking infrastructure. The user behavior model contains several aspects that can best distinguish normal activities from attacks. These aspects contain failed login attempts, login access patterns, and user behavior. Researchers have shown that the IDS with log analysis approach effectively identified potential intrusions and misuses with an acceptable overall alarm rate. Continuous tuning of the intrusion detection model can be used to enhance the accurate detection rate, and Yu et al. [74] has presented a case in point. The paper introduces the ADAT IDS that uses a buffer for holding suspicious predictions. Only the predictions whose confidence values are higher than predefined thresholds are sent to the system operators for identifying false predictions. The feedback is then used to tune the detection model so that misclassified attacks are dropped resulting in higher detection accuracy.

From the literature review, we can summarize that an IDS that quickly detects and classifies known and unknown attacks with a high detection accuracy is one of the most essential components of a SPHIS. The output of this module provides the security state of the system. If the system is compromised, network forensics analysis must run to learn unknown attack signatures, followed by internalizing the necessary responses needed to recover system performance and defend against such attacks.
5.2 Network forensics analysis (NFA)

Network forensics aim at finding out causes and impacts of cyber attacks by capturing, recording, and analyzing of network traffic and audit files [75]. NFA helps to characterize zero-day attacks and has the ability to monitor user activities, business transactions, and system performance. As a result, attack attribution, what attack methods and tactics were used and attack duration, can be analyzed and derived using NFA.

Recent articles claim that self-protecting systems are able to mitigate attack impacts by “static” IRS. Protection mechanisms selected by this type of IRS remain the same during the entire attack period [76]. None of these self-protecting systems, however, contain NAF. Notwithstanding, a static IRS is limited in its ability to fully and efficiently eliminate these attacks. To establish a fully autonomic computing system, as described in Section 2.2, the NFA must be activated once unknown attacks have been identified by the IDS. Some recent research performs the NFA as a postprocess analysis, but live NFA is required to support the 24/7 system availability. In this way using live NFA, the investigation of unknown attack signatures will not disrupt normal client requests. Additionally, live NFA is executed immediately after unknown attacks are detected, which reduce latency and quickly closes the window of vulnerability.

Gunter [77] discussed research on balancing access controls with audit in medical record systems, and using audit logs to understand roles and workflows to develop reliable anomaly detection and safe access controls. Philli [75] illustrated a generic process model for network forensics in real-time and postattack scenarios. The first phase is preparation, which makes sure security solutions such as monitors, firewalls, and IDSs have already been deployed. The second phase is detection, which identifies cyber attacks and captures network traffic, followed by the third incident response phase. The fourth phase is collection and preservation. Traffic data are collected and preprocessed for storage on backup devices for presentation. Together these phases identify attack indicators, classify attack patterns, determine attack paths, and provide documentation offline.

The challenges of next generation state-of-the-art NFAs are described by Philli [78]. The analytical complexity of real-world large-scale HISs grows exponentially, therefore, making full NFA computationally infeasible. In some NFA scenarios accurate vulnerability states are not available by virtue of the fact that the NFA process is fundamentally postcompromise. Also, analyzing an extremely large amount of auditing data in a very short time span is computationally infeasible. Additionally, in NFA it is often very
difficult to distinguish primary attack signatures from trivial signatures. Let us examine today’s live NFA tools.

Investigating high-volume auditing data in real time depends primarily on the abundance of computational resources. Normally, the size of one-days worth of auditing files for large-scale systems is larger than 100 GB. Data concentrators are used, for example, by data historians within the context of the distribution systems (i.e., the smart grid). Cloud storage platforms are, used to store and process such data. Chen [79] used the cloud storage and computing platform to analyze offline phishing attacks. Phishing filter functions can effectively scale to detect phishing attacks. Research described in [80, 81] adopts the Hadoop map reduce model to analyze a high volume of log files and extracts attack signatures for intrusion detection and prevention.

Contemporary NFA approaches manually investigate attack patterns after HISs are compromised. Real-time and dynamic forensics systems, which investigate attack causes and impacts automatically, have been deployed to address the scalability issue of such manual processes. The utilization of the cloud-based approaches makes NFA more practical to deploy. Nonetheless, the current question about how to realize a dynamic and automatic NFA system is the subject of much research. For example, Wang et al. [78] developed immune agents, which automatically generate crucial evidence of unknown attacks for rapid active responses. An automatic method was proposed by Wang et al. [82] to make regular expression signatures of the HTTP attacks offline. This approach first extracts application session payloads to identify common substrings and their positional constraints. Using these extracts, attack signatures are transformed into regular expressions.

Li et al. [83] designed a network-based signature generator, namely, LESG. The generator automatically analyzes zero-day buffer overflow vulnerabilities and generates length-based signatures via a three-step algorithm. The generator selects field candidate signatures, optimizes the signature lengths, and derives the optimal signature that produces minimal false alarms. The LESG has lower computational overhead than research by Brumley et al. [84], which generates vulnerability signatures using regular expressions. False alarm rates of an IDS applying the LESG are significantly reduced and the attack detection speed is improved. The LESG is also tolerant to noisy traffic.

The utilization of NFA allows the self-protecting system to investigate and learn the causes and impacts of an unknown attack. In the end, the HISs can be protected from both known and unknown attacks autonomously (i.e., without human intervention).
5.3 Intrusion response systems

An intrusion response system (IRS) is a critical part of the self-protecting system for ensuring appropriate responses are dispatched to react to protect the HIS and recover system performance back to normal. The tight interaction between the NFA module and the IRS module makes future attacks, which have similar signatures, less likely to succeed [85]. The development of an autonomous IRS focuses on two key elements: configuring suitable responses and evaluating recommended responses dynamically.

Responses are protection mechanisms that have the ability to eliminate or mitigate cyber assaults. Protection mechanisms, which are configured and implemented to defend the system against specific cyber attacks, vary according to the types of cyber attack they target. In other words, recent research shows that using key-agreement schemes and a heart-to-heart authentication system are meaningful approaches to secure implantable medical devices (IMDs) [86, 87]. However, these responses are not efficient in removing DDoS attacks or SQL injection attacks. Five categories of cyber attacks are illustrated in Section 4.2: scanning attacks, spoofing attacks, injection attacks, broken authentication and session management, and DDoS attacks. In the subsequent paragraphs, responses that protect HIS web applications from these attacks are reviewed.

Scanning attacks are conducted by adversaries to discover vulnerable services. Port scanning, IP addresses scanning, and version scanning attacks are discussed in Section 4.2. Techniques such as TCP scanning, UDP scanning, and ICMP scanning gather the IP addresses of victim hosts looking for open ports. To protect HISs from scanning attacks, packet filters are placed in the victim system to discard scanning packets. Transparent proxies, which hide actual open ports from external users, are also somewhat effective mechanisms to defend the system from scanning attacks. A successful spoofing attack will utilize legitimate identity credentials to enable or bypass authentication mechanisms and unauthorized remote system access. Securing Internet protocol communications by authenticating and encrypting IP packets is a promising defense measure (e.g., using the Internet protocol security (IPsec)). Another example is secure socket layer and transport layer security (SSL/TLS), which uses public key infrastructure to secure transport layer transactions between clients and servers [88–90]. Mechanisms like secure remote password protocol and dynamic security skins are generally used by transaction based web sites to defend against authentication attacks [91, 92]. In addition, both access control lists, which block communications
between malicious clients and servers, and packet filters, which reject spoofing attack packets are useful spoofing countermeasures.

Injection attacks insert illicit data into web applications and force victim systems to execute undesirable codes or commands and spread malware. One of the best methods to protect from injection attacks are “secure coding best practices.” The Open Web Application Security Project (OWASP) lists 12 secure coding practices such as input validation, output encoding, and file management checklists [93]. Limiting web application coding privileges, reducing debugging information, and testing web applications regularly can also mitigate injection attacks [94].

Brute force attacks and man-in-the-middle attacks are broken authentication and session management attacks, which exploit flawed credential management functions. Once attackers successfully break authentication mechanisms or hijack active sessions, it may be possible for them to gain access to all accounts and do anything legitimate users could do. To protect against broken authentication and session management attacks, a simple but efficient method is to block IP addresses. Another simple and effective practice is to create and follow security guidelines for choosing strong passwords. No default usernames and/or passwords are allowed. Disabling remote logins via the secure shell and creating a large number of honeypot accounts in financial systems is a common practice used to mitigate brute force attacks [95–97].

Flooding-based DDoS attacks as discussed above (Section 1 and Section 4), aim at impeding legitimate clients from accessing host/server resources. There are numerous traditional mechanisms the host can employ to defend against DDoS attacks. Mudhakar et al. [98] used a twofold mechanism to filter application layer DoS attacks. This approach uses hidden ports to filter illegitimate packets. Then web server resources are allocated to admit clients based on priority levels assigned by the congestion control element. This protection mechanism works well for external DDoS attacks, but it is ineffective in eliminating internal DDoS attacks (i.e., compromised internal nodes initiate the flood). Performance overheads from applying this approach are very high resulting from the necessary adjustments needed to employ hidden ports and to monitor/manipulate client priority levels.

Stavrou et al. [99] introduced WebSOS, an adaptation of the Secure Overlay Services (SOS) architecture that only allows legitimate users to access web servers when DoS attacks have compromised the server. Patil and Kulkarni [100] proposed a lightweight mechanism that uses trust to
differentiate between legitimate users and attackers who launch HTTP flooding attacks. However, it is unclear whether this mechanism will reduce high utilization of web server resources caused by application layer DDoS attacks.

The first key element of an IRS is selecting and configuring protection mechanisms. Any of the above mentioned mechanisms can eliminate or at least mitigate the variety of potential attack tactics. The second element is a way to evaluate these candidate response mechanisms for choosing the most appropriate mitigation/recovery response. The most effective response, however, may not protect a system from other attack types. To accurately evaluate these responses, Luo et al. [101] designed a game-based multistage intrusion defense system, which allows administrators to evaluate which type of responses are optimal with respect to different measures (i.e., selectivity, rapidity).

The optimal responses are the ones with the highest utility values based on administrator estimations of expectations and variance. Another useful evaluation method is to employ cost-sensitive modeling to select optimal responses. This method accounts for the balance between an estimated cost of potential damage vs the cost of implementing a particular set of responses [102, 103]. Chen et al. [33, 34, 44] developed a multicriteria analysis controller for the IRS to select the most appropriate responses consider their protection functionality and overheads. The researchers demonstrated that their approach could react to various cyber attacks to distributed systems and SCADA systems, and the IRS regulated system behavior back to normal in real time.

Dewri et al. [104] addressed a multiobjective optimization problem for how to secure systems with limited budgets. The researchers created a system attack tree model to evaluate measures based on total security control costs and residual damages. This approach provides concrete options to the system administrators about the quality (i.e., cost) of the different trade-off solutions. However, system administrators are not allowed to modify their decisions during run-time. Stakhanova et al. [105–107] evaluated finite responses for three criteria: response system impacts, response goodness, and response operational costs. Their response evaluation relies on expert knowledge, but the evaluation cannot distinguish which action is the best if all responses have similar descriptions.

Designing dynamic IRSs must account for installing and configuring suitable protection mechanisms as well as selecting corresponding
evaluation methods. The dynamic response system is an essential module in a self-protecting system that actively reacts to (thwarts) attacks after they are detected by the IDS. This approach reduces the time gap between attack detection and response, which reduces the (time) window of opportunity for the attacker. Iannucci et al. [108] present a dynamic IRS using Markov decision process (MDP) to evaluate its long-term response policies according to a multicriteria objective function. The proposed IRS was validated to protect large systems while introducing little or no overhead on the protected hosts.

Additionally, to protect HISs against cyber attacks is not limited to the technology aspect but also should include security laws, regulations and guidelines as well as establishing a comprehensive reporting system to capture security-related failures in medical devices [109].

6. Conclusion

This book chapter first reviewed the history of healthcare information systems (HISs), introduced and analyzed recent emerging cyber breaches that affected HISs, and then proposed the autonomic computing concept and applied the concept to design self-protecting HISs (SPHISs) that can defend themselves against cyber intrusions with little or no human intervention. To realize such SPHISs, we studied security vulnerabilities residing in HIS networks and the HL7 communication standard, presented a feasible design, and discussed the functions of each component in detail. The main components of a SPHIS should contain monitoring systems, early estimation modules, intrusion detection, network forensics analysis devices, and intrusion response systems. Although there is a gap in current research for developing SPHISs, we surveyed the self-protecting systems and their approaches designed for similar computing systems such as enterprise systems and industrial control systems. We also reviewed and evaluated recent research on each component of the SPHIS. This chapter gives readers an idea how to develop their own self-protecting systems for specific healthcare IT environments to reduce human’s burden while enhancing the security posture.

Acknowledgments

This study is supported by supported by the National Science Foundation (NSF) under Grant No. 1812599. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the NSF.
Toward realizing self-protecting healthcare information systems

References


[70] B. Amann, R. Sommer, A. Sharma, S. Hall, A lone wolf no more: supporting network intrusion detection with real-time intelligence, in: Research in Attacks, Intrusions, and


About the author

Dr. Guenevere (Qian) Chen is an Assistant Professor with the Department of Electrical and Computer Engineering at the University of Texas at San Antonio (UTSA). Before joining UTSA, Dr. Chen was an Assistant Professor and Coordinator of the Computer Science Technology Program at Savannah State University. She earned her Ph.D. degree in Electrical and Computer Engineering from Mississippi State University in 2014.

Dr. Chen’s primary research area is autonomic computing and cyber security. She developed an Autonomic Security Management framework that proactively defends the computing systems against known and unknown cyber attacks with little or no human intervention. The ASM framework has been successfully applied to secure distributed systems, industrial control systems (e.g., SCADA), high performance and cloud computing and the Internet of Things (IoT) ecosystems. Her research interests include risk assessment, malware behavioral analysis, early warning intrusion detection and response, and end-to-end security solution development.
This page intentionally left blank
CHAPTER FOUR

SSIM and ML based QoE enhancement approach in SDN context

Asma Ben Letaifa
MEDIATRON Lab, SUPCOM, University of Carthage, Tunis, Tunisia

Contents

1. Introduction and challenges 153
2. About QoS/QoE parameters 155
   2.1 QoS 155
   2.2 Intrinsic QoS 155
   2.3 QoE 156
   2.4 User experience 158
   2.5 Correlation between QoS parameters and QoE 158
3. Environment and challenges 159
   3.1 SDN: Software defined networking 159
   3.2 QoS and QoE context 161
   3.3 How can SDN controllers manage the QoS/QoE? 162
   3.4 QoE challenges in SDN environment 163
   3.5 MSN, CDN and content-centric networks in big data context 164
4. QoS and QoE metrics in different services applications 165
5. How modeling QoS/QoE in related work? 166
   5.1 Video streaming overview 167
   5.2 Modeling QoE in a video streaming context 167
   5.3 Mathematical background 168
6. Video streaming protocols 168
7. Enhancing QoE approaches 175
   7.1 Machine learning approach 175
   7.2 Adaptive coding 177
   7.3 Flow classification and monitoring QoE 177
   7.4 Buffering impact 178
   7.5 Path optimization 178
8. Simulation environment and testbed 178
   8.1 Testing on Mininet 179
   8.2 Graphical analyze 179
   8.3 Simulation scenarios 180
   8.4 Interpretation 181
Abstract

Today, video streaming rose above all other traffic types over the internet. While providing this service with a high quality is the most challenging task, researchers are trying to solve the challenge by giving a more efficient network where congestion, broadband limitations and unsatisfied users are limited. In new multimedia based networks, new challenges move from technology-oriented services to user-oriented services which prove the importance of QoE. Service providers’ growth depends nowadays not only on QoS parameters but also on clients’ feeling and expectation. That is why; service providers must measure received QoE and this becomes a challenge regarding the evaluation of users’ feeling. For users, qualitative perception differs from one user to another and service providers affront difficulties to transform the qualitative values into the quantitative one. The QoE evaluation needs more sophisticated methods to describe the real expectation of users. New multimedia applications provide at any user locations media who are sharing video, and communicating together in virtual network. These applications require providing the best possible QoE to consumers. When it comes to us, we present in this paper a machine learning approach combined with adaptive coding in order to provide a better QoE for video streaming services. This solution will be established using SDN architecture. We can justify this choice because we need a centralized architecture, where the totality of the network is known, to predict its status. So, we will implement a machine learning algorithm in the controller: this algorithm, called ML-based SSIM, will calculate approximately the quality needed for a video to be streamed. Finally, the quality found by the ML-based SSIM Algorithm will be combined with the network situation to choose the right coding. First part of the paper deals with an introduction of QoE requirement, metrics and protocols used especially in streaming services, then we give a complete study around Machine learning algorithms and other fields used in literature to enhance QoE. We define in this paper, at first, QoS and QoE, then the serving environment such as mobile cloud computing and software Defined Network (SDN). Then, we give both objective and subjective metrics, expose mathematical approaches used in modeling, predicting and evaluating QoE. We define in this paper, at first, QoS and QoE, then the serving environment such as mobile cloud computing and software Defined Network (SDN). Then, we give both objective and subjective metrics, expose mathematical approaches used in modeling, predicting and evaluating QoE. Second, we expose the SSIM approach and explain how our proposed one is based on. The last part of the paper deals with experiments: we describe SDN environment deployment, describe scenarios and finally simulate on SDN emulator some topologies to demonstrate the impact of SDN components helping QoE measurement. At the end, we give the results and values. We highlight the future of our proposition.
1. Introduction and challenges

New services such as data storage or processing are outsourced to the Cloud (e.g., Google Apps, or Cloud gaming or streaming on YouTube), and are making users more and more dependent on the network to keep with their daily activities. A metric, called quality of experience (QoE) is appeared to measure user’s satisfaction on services done on the internet networks. QoE is a research paradigm to quantify the users’ perception on consumed services. One of the most used services is online video streaming, which, in this last decade, has seen a huge growth. The video traffic on internet represented 66% of all global Internet traffic in 2013, and will be about >79% of all Internet traffic by 2018. The evaluation of QoE is based on metrics such as startup time, average playback bitrate, etc. It’s providing a better indication of the satisfaction of the clients whose perception is subjective. They have to always consume reachable and react fast services, visual artifacts should not affect their video streams and audio impairments haven’t to affect voice calls. Network optimization was traditionally focused on optimizing network properties such as QoS parameters, but these years, QoE depends on both users’ perception and quality consumed service, they form end-to-end metrics. We argue that QoE is the measure that is relevant for network operators and service providers.

At the other hand, for current networks, especially mobile social networks (MSNs), providers face a new challenge to provide mobile social user with a better QoE. With new networks generations like 5G, the next network architecture beyond the current 4G, operators and service providers aim to offer low latency connectivity, enable smart and programmable nodes for the Internet of Everything, and adopt the new functions of Software Defined Networking (SDN) and Network Function Virtualization (NFV). The main SDN concept [1] is that both control and data functions are decoupled. New research studies are discussing various challenges posed by SDN, such as virtualization and cloud service applications, networking with other platforms, design of switches and controller, etc. SDN has the potential to facilitate the deployment and management of network applications and services with greater efficiency and beneficiate from a new centralized control mechanism.

Multimedia services know a rapid growth today and become the principal traffic source on the internet. Traditionally, service providers and network operators were facing the challenges to have a higher bandwidth and
satisfy multimedia traffic Quality of Service (QoS) requirements. But nowadays, those requirements are fast changing and the actors (providers and operators) are facing new challenges about the quality of audio and on real-time multimedia applications, like video streaming ones. Recently, with the variation of multimedia applications, and in order to guarantee the best QoE to their clients, service providers and operators see emerging new networks architectures and management policies. With those new multimedia applications, new networks called cloud and mobile cloud computing are emerging and represent an efficient environment to provide mobile services for daily life. QoE on mobile devices plays an important role in Mobile Cloud computing MCC and new network topologies, than the other devices properties such as limited hardware resources, mobility, etc. MCC is so different from cloud computing. We argue also that new architectures are defined and we encounter NFV and SDN context, which came with virtualization paradigm and offer a good QoS/QoE with this new concept.

On the other hand, the QoS is defined as a “metric represented by some objective service parameters that can be effectively measured based on the traffic analysis (such as time delay, packet loss rate, etc.).” Moreover, when dealing with Mobile Cloud, SDN networks or IoT environment, a variety of services is considered and just QoS doesn’t help enough to satisfy traffic analysis and we need to express a metric called QoE to evaluate the quality. That is why; the QoE metrics is more involved with multimedia services, since it measures the user expectation. The last years, video streaming services represent the main source of generated traffic over the internet. As we have mentioned, although the QoE is used to measure the end-user satisfaction; the problem of its evaluation is much more complicated than the QoS evaluating problem already done in traditional networks with usually objective methods. In other words, in order to evaluate QoE, it is necessary to know how much an end-user would like/dislike the delivered service. Thus, the problem arises of modeling the user perception in machine learning field or mathematical area where both objective and subjective methods are used for this purpose.

When it comes to us, we present in this paper a machine learning approach combined with adaptive coding in order to provide a better QoE for video streaming services. This solution will be established using SDN architecture. This paper deals with an analysis of QoS and QoE challenges [2–5]. Hence, we start by providing an overview of new networks topologies and QoS/QoE definitions and metrics (both objective and subjective),
so we focus on extracting relevant key performance indicators for QoE evaluation, then we study mathematical background helping to model QoE especially in the context of Video streaming services. In this paper, we describe at first the study context, we will introduce QoS, QoE, SDN networks, and machine learning fields, and then we describe SDN and make a large overview on QoE metrics and streaming protocols. After that, we present our methodology for service quality estimation and validate it with some experiments: that is why we study different topologies made on SDN networks, some scenarios helping to offer best QoE. The main goal of those scenarios is to analyze the impact of network conditions on video streaming services. Next, we expose our approach to enhance QoE in SDN context with a modified approach based on SSIM metric, we argue our choice and give results of experiments done on streaming service video. At the end of this paper, we conclude the work by giving conclusions, future work and perspectives.

2. About QoS/QoE parameters

Operators and Service Providers must manage the quality to offer service guarantees to their end-users. QoS and QoE represent metrics used in quality of service evaluation. Traditional metrics are already used to estimate the new user QoE when measurements are perceived, by the end-user. Furthermore, QoE measurement is too long to be collected by network measurements. The challenge is to obtain real time QoE measurements at any network node and then to proceed to real time traffic management process.

2.1 QoS

According to the ITU, QoS is defined as: “The totality of characteristics of a telecommunications service that bear on its ability to satisfy stated and implied needs of the user of the service.” Delay and percentage of lost packets known as traditional parameters are helping to evaluate the QoS. But we consider that those parameters don’t reflect the expectation client. That is why; QoE is more used recently because it reflects really the user’s satisfaction degree.

2.2 Intrinsic QoS

Objective parameters such as delay, jitter, and loss are making Intrinsic QoS which represents the network-oriented QoS and constitute the major
problem of services providers and operators [6]. Latency or delay, defined as an amount of time that a data packet needs to transfer between two endpoints, jitter, defined as an undesired deviation of latency between two consecutive packets, packet loss, (or lost packets percentage during transmission, bandwidth or amount of available bandwidth represent the main four parameters evaluating the QoS over the network [7]. According to recommendations in Ref. [8], these QoS parameters can be examined from three points of view (1) QoS of the client, (2) QoS that provider is offering, (3) QoS that a client perceives. However, we define Perceived QoS as the quality perceived by the users.

2.3 QoE

QoE is defined in Ref. [8] as “the overall acceptability of an application or service, as perceived subjectively by the end-user.” Recommendation tells also that “it includes the complete end-to-end information about client, terminal, network infrastructure; and may be influenced by the client context.” Hence, measuring a subjective QoE may differ from one client to another, it is usually estimated using objective parameters. The relationship between QoE and QoS is non-trivial and we have to analyze if additional factors can influence the perception of quality for delivery of multimedia content especially in cloud computing environment and SDN networks. Fig. 1 shows factors contributing to QoE.

Subjective QoE metrics quantify the perceived quality of a service by the end-users. Hence, QoE needs to gather numerous parameters such as the encoding, transport, content, type of terminal, network, services infrastructure, media encoding, etc. as well as the user’s expectations [9]. Some major multi-dimensions like effectiveness, usability, efficiency, expectations and

![Fig. 1 QoE contributing factors.](image-url)
context are also affecting QoE. Moreover, those subjective QoE concepts are expensive and time-consuming since it requires human participation. Authors of Refs. [10,11] tell us that MOS is representing the QoE but says also that QoE assessment can be subjective (qualitative) and objective (quantitative). Subjective methods are based on people participation, when objective methods based on several parameters measurement.

### 2.3.1 Objective parameters

For quantitative QoE measurement, we can found three models at literature: No-reference (NR), Reduced-reference (RR) and Full-reference (FR). Table 1 summarizes most popular objective video quality estimation methods in this context.

### 2.3.2 Subjective parameters

Table 2 gives a survey to main subjective parameters for both speech and multimedia traffic quality measurements.

By providing mathematical models for the quality estimation, Objective methods try to overcome limitations caused by subjective ones, but known models do not consider yet other context parameters (terminal, location, time-of-the day, etc.). That is why, we think that we have to propose a

<table>
<thead>
<tr>
<th>Table 1 Objective video quality methods.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NON Human Visual System (HVS) perception</td>
</tr>
<tr>
<td>Peak Signal-to-Noise Ratio (PSNR)</td>
</tr>
<tr>
<td>Poorly correlates with the human perception of visual quality</td>
</tr>
<tr>
<td>Video Quality Model (VQM)</td>
</tr>
<tr>
<td>Perceptual effects of video impairments. A single metric that combines blurring, jerky/unnatural motion, global noise, block and color distortion</td>
</tr>
<tr>
<td>Perceptual Video Quality-of-Experience Measurement (PEVQ)</td>
</tr>
<tr>
<td>Degraded video signal is compared to the original signal on a perceptual basis</td>
</tr>
<tr>
<td>Human Visual System (HVS)</td>
</tr>
<tr>
<td>Moving Picture Quality Metric (MPQM), Motion based Video Integrity Evaluation (MOVIE), Perceptual Quality Index (PQI)</td>
</tr>
</tbody>
</table>
hybrid method, like what is done in INRIA with PSQA model and insert parameters such as environmental, personal, social, cultural, technological and organizational class.

2.4 User experience

According to Mayer [12], the user experience is classified thank to four measures: perception, rendering quality, physiological, and psychological. According to authors of Ref. [14], for tele-immersive services, no prediction is available, such as World Opera application or collaborative gaming. However, prediction models exist for interactive multimedia services such as VoIP. The evaluation of objective QoE is strongly related to the network, the QoS metrics are really used to calculate this QoE value. Difference between both evaluation methods (about network/human) may cause that providing the best QoS do not necessarily provide the best QoE.

2.5 Correlation between QoS parameters and QoE

QoS metrics are generally related to device including CPU, memory use, packet loss, delay or jitter. When QoS focuses on the equipment’s controllers or network traffic, QoE performance indicators are user-centric (for instance: webpage loading time, application response duration, VoIP MOS) and reflect the user’s satisfaction and feelings. The relationship between QoS and QoE is not linear so correlation is hard to estimate. We can find in literature a variety of QoS/QoE correlation algorithms. In Ref. [14], we describe, for example, the relationship between the QoS and QoE metrics, including

<table>
<thead>
<tr>
<th>Table 2 Subjective parameters.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean opinion score (MOS) [12]</td>
<td>For speech quality and multimedia traffic average human rating (say 1–5 scale)</td>
</tr>
<tr>
<td>Double Stimulus Continuous Quality Scale (DSCQS) [13]</td>
<td>Subjective video quality evaluation</td>
</tr>
<tr>
<td>Double Stimulus Impairment Scale (DSIS)</td>
<td></td>
</tr>
<tr>
<td>Single Stimulus Continuous Quality Scale (SSCQS)</td>
<td></td>
</tr>
<tr>
<td>Single Stimulus Continuous Quality Evaluation (SSCQE)</td>
<td></td>
</tr>
<tr>
<td>Absolute Category Rating (ACR)</td>
<td>Video quality instantaneously</td>
</tr>
<tr>
<td>Stimulus Comparison (SC)</td>
<td></td>
</tr>
</tbody>
</table>
media type or another algorithm that describes QoE evaluation algorithm adapted to a human’s brain in order to “predict” number of like/dislike. Some works found in literature describe also several QoS metrics that impact overall QoE: We can simply predict QoE, with appropriate assumptions based on QoS measurement or we just deduce the net required service layer performance based on appropriate QoS assumptions. Several correlation models for Video on Demand (VoD) service or web video estimation models are studied.

3. Environment and challenges

3.1 SDN: Software defined networking

SDN architecture presents a new solution that consists of separating the control plane from the data plane which is typically coupled together. Network functions traditionally realized in specific hardware can now be abstracted and virtualized on any equipment. A split between control and data path nodes is performed, so a centralized controller has a global view of the network while the data plane includes devices which simply forward packets following rules expressed by the controller. In order to communicate between these two layers, an open standard protocol is employed. This separation between the two layers simplifies the network management and help to simply program network control.

The main concept of SDN architecture consists on the separation between control and forwarding functions. The Figs. 1 and 2 demonstrate the multiple components of an SDN architecture which is based on three layers separated by open interfaces. The SDN application plane is a layer composed of a variety of applications that communicate via Northbound APIs. It’s responsible for management, reporting functionalities such as monitoring or security. The SDN controller plane represents the main entity in the network that facilitates the creation/ destruction of network paths. Typical SDN Controllers are OpenDaylight and Floodlight. The SDN data plane includes different devices deprived from any intelligence. They simply execute the controller’s rules. The SDN architecture defines also the key interfaces between the different layers, which are East/West bound API that are implemented by the different controllers of the SDN and used to facilitate communications between them. Hyperflow is one representative example of such APIs. Southbound API is implemented by the different forwarding devices in the SDN and enabling the communication between these devices and the controllers. For such APIs, we can enumerate OpenFlow or NetConf.
Fig. 2 SDN architecture and topology.
The *Northbound API* is implemented by the controllers of the SDN and used to facilitate the communication between controllers and the network management applications. In such SDN architecture, enhancing the QoE became easier by implementing specific algorithms within the controller entity. The SDN manages in such case to enable new functionalities such as QoE monitoring and enforcement functions.

### 3.2 QoS and QoE context

Quality of Service is defined by the ITU as the “totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs.” It defines also other metrics helping to providing guarantees in term of delay, jitter, and packet loss that are useful for user satisfaction.

For example, ITU-T Recommendation G.1010 states that for interactive voice communication the delay should be below 150 ms and the packet loss rate should be below 3%. In other hand, QoE is defined as “the degree of delight or annoyance of the user of an application or service. It results from the fulfillment user expectations with respect to the utility/enjoyment of the application or service in the light of the users’ personality and current state.” QoE aims to capture the users’ perception when using a service. To address this challenge, we review in this paper the most common quality metrics especially for video streaming services. These metrics estimate a set of influence factors impacting QoE. We illustrate in Table 3 an overview of relevant metrics, a selection of what we consider to be the most relevant and often used QoE metrics for speech, video, and web browsing. For an

<table>
<thead>
<tr>
<th>Table 3 Overview of QoE metrics.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Speech</strong></td>
</tr>
<tr>
<td>No-Reference NR</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Reduced-reference RR</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Full-Reference FR</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
extended overview of available metrics, we refer to Ref. [15] for speech quality, to Ref. [16] for image quality, and to Ref. [17] for video quality. QoE metrics can also be classified into three categories by the required amount of reference information. Full-reference FR metrics estimate the QoE score based on the original signal and received perturbed signal. No-reference NR metrics account for this challenge and estimate QoE_score purely based on the received signal. Reduced-reference RR metrics account for this inaccuracy and estimate it based on a subset of features.

The challenge on QoE is encountered since the users’ perception is subjective. The main objective of QoE is to quantify the users’ perception of the applications from service generation, including transport entities until the end device’s screen or audio unit. Both User satisfaction and service qualities are strongly correlated from end-to-end serving entities.

3.3 How can SDN controllers manage the QoS/QoE?

SDN allows the evolution of mechanism and service deployment on programmable networks. Hardware and control unit are decoupled in SDN. Network intelligence is centralized in controllers and the other nodes are reduced to simple flow forwarding entities that can be programmed via interfaces. Openflow is the standard protocol that acts as interface between the Network Controller and the network entities [18]. Some related works that offer QoE monitoring done all over SDN are described in Ref. [19]. Other papers focus on the design of video control planes and consider the use of resource allocation based on video quality [20] or QoE [21]. Since the QoE is not directly measurable, network operators face the challenges to calculate a QoE Key Performance Indicator KPI value from measurable QoS parameters. We are facing three assumptions: (1) the network controller has a global view of the network, (2) it monitors others network nodes, (3) openFlow does not support QoS configuration, we argue that we can easily implement a new rule or a new module in the controller node and implement it to enhance QoE. For example, authors of Ref. [18] propose for this case two approaches: (i) the first one by allocating network bandwidth slices to video flows and (ii) the second one by guiding video players in the video bitrate selection. They implement the open Day light controller through two communicating HTTP servers, one hosted by the controller machine and one hosted by the switch. The HTTP server hosted at the controller maintains the Active Flows Table, executes the Optimization Module in a Python thread, and establishes communication pipes; they prove that
several QoE metrics, such as Video Quality Fairness, video quality and switching frequency can affect SDN performance. In another work [1,22], authors used RYU controller to receive per-client QoS policy messages coming from the application layer. The SDN-controller here passes these messages to the internal SDN-based application (implemented inside the RYU controller responsible for resource allocation, monitoring and QoS provisioning) to configure, allocate, provision, and monitor the network resources for each client accordingly. In different framework Sangeeta et al. [23] proposed a model where all the intelligence and the administration of the work are located in the SDN network, their ultimate goal is to optimize the overall QoE. To estimate the available bandwidth correctly, the proposed optimization application is based on several metrics such as device type the network topology which are obtained from the SDN controller swiftly and easily.

In one word, in all these research works, the role of the controller is clear and very important where it facilitates the communication, the management and the administration of the whole architecture, and the addition of the new rules of QoE optimization.

To fast deploy and test new services, applications and protocols, we can test scenarios on an emulation and simulation tools named Mininet. We can also use ns-3 network simulator which supports OpenFlow switches.

3.4 QoE challenges in SDN environment

We find in literature some proposals answering this QoE challenges. SDN/NFV based solution for QoE monitoring in mobile networks developed in “ISAAR project” proposes in this purpose three functional components: QoE Monitoring (QMON) for flow detection and assessment, QoE Rules (QRULE) for policy rules and permission checking and finally QoE Enforcement (QEN) for respective flow manipulation. We argue that SDN/NFV augments the existing QoE monitoring and enforcement. That is why described project ISAAR makes use of SDN to selectively copy out flows as well as to enforce flow manipulation actions. It goes throw functional block structure (QMON, QRULE, QEN) allowing direct NFV implementation. Those three function split options have been presented: SDN only, SDN + NFV “light” and SDN + NFV “full” with good QoE results. On the other hand, paper [24] introduces an in-network QoE Measurement Framework (IQMF) that provides live network-wide QoE measurements. The framework manages the network without any interaction.
with client. For such purpose, IQMF adopts quantitative metrics to measure the video fidelity QoE and representation impact.

Several researchers have investigated Mobile context with SDN. Zafar et al. [25] focus on mobile application detection. They use a ML-based traffic classification method in order to identify flow types in SDN. Ali et al. [26] used OpenFlow to cope with MPLS-TE and MPLS VPNs. Saurav et al. [27] argue aggregation engineering in a packet circuit network [28]. Michael et al. [29] have investigated on YouTube application.

Operators using 3GPP QoS mechanism address service flow management. An overview streaming application based on HTTP is found in Ref. [30]. Other approaches also address applications enhancements. They define HTTP Adaptive Streaming Services (HAS) [31] as a new way to keep with the video streaming quality. Other approaches are about sharing contents on Fixed-Mobile Convergence (FMC) [32]. The ISAAR framework presented in Ref. [33] follows a different approach. It aims for service flow differentiation without PCRF support or PCRF based flow treatment.

Video streaming services have grown to become increasingly popular over the recent years. Enhancing video delivery and clients QoE, Kim and Choi [34] proposed the Server and Network Assisted DASH (SAND) architecture. SAND is a control plane for video delivery that obtains QoE metrics from the users (clients) and returns network-based measurements to help the clients enhance their overall QoE. The authors of Ref. [35] propose a measurement approach for YouTube video traffic. QMON [36] enhances previous work by supporting more video codecs. Specifically, it estimates the playout buffer levels [37] and the number of stalls and their duration, and then calculates the MOS (Mean Opinion Score) for YouTube videos.

3.5 MSN, CDN and content-centric networks in big data context

We assist nowadays to the explosion of mobile social networks (MSNs) which provide users with various social applications who have disposition to obtain data on mobile devices instead of their traditional desktop computers. In this context, Mobile Data are delivered over Content-Centric Networks CCNs [38] to settle new challenges (1) we can store a very large volume of mobile big data by using the cache space in the content store, (2) we can manage fast mobile social data exchange by using named data and interest packets, (3) we can deliver variable mobile big data with corresponding agents nodes selection. To face with those challenges, future works have to answer these three questions: (1) How to control dynamic
mobile big data? (2) How to make an efficient resource allocation for mobile big data? (3) How to guarantee privacy in case of multiple replicas?

CCN or differently named Information-centric networks ICN allow the network to cache data by attaching storage to routers. Recent works consider different aspects of video streaming and rate adaptation. The work of Adhikari et al. described in Ref. [39] is performing measurements to evaluate Netflix. The authors of papers [16,40] study dynamic rate adaptation for streaming services and facilitate the distribution of video. A new field is appeared these last years: Big data in MSNs which can be described as mobile big data, which have large volume, wide variety, fast velocity, and economic value. Different from conventional big data, mobile social big data should have in particular the following characteristics [38] Volume, Velocity, Variety and Value. Within MSNs, mobile big data have a wide variety of data, including video, audio, and file sharing. We give here some statistics taken from Ref. [38]: In 2015, mobile data traffic caused by video is 2,399,765/mo; mobile data traffic of other file sharing is about 74,694/mo. Besides, compared to 4.2 EB/mo of mobile data traffic in 2015, mobile data traffic is predicted to ranch 24.3 EB/mo in 2019. With this growth, new challenges, with new QoE requirements, are faced: (1) better services are hoped to be provided for mobile social users with a satisfactory (QoE). (2) Storing mobile big data will face new difficulties due to the limited capacity of mobile storage in MSNs. (3) Mobile big data should have a fast rate to match the speed of data production. (4) The variety of mobile big data should be considered to satisfy the demands of different mobile social users. (5) Mobile big data should be studied with consideration of their value. Research studies have to discuss new challenges in MSN: How to store a very large volume of mobile big data in MSNs? How to manage mobile social data with fast velocity? How to deliver mobile big data with variety? How to control mobile big data based on value?

4. QoS and QoE metrics in different services applications

Network, content and video service providers take advantage of content distribution networks (CDNs) and would like to ensure a high degree of video QoE for their clients. We give in Table 4 the most used applications on data networks and explain after the main used metrics for each traffic.

Table 3 provides a brief description for multimedia quality metric as they had been used in literature.
Over the years, multimedia applications have conquered several segments of the telecommunications field. Today, we are dealing with multimedia services in many areas, starting with different digital television systems (e.g., DVB), video telephony, video on demand (VOD), Internet Protocol TV services (IPTV), the voice over IP (VoIP) or simply, video sharing services like YouTube or Dailymotion. The development of these services and the end-to-end optimization of these systems are closely related to the perception of quality by the user and his satisfaction with the service rendered. In this sense, there is a deep need for a measure that reflects the satisfaction and perception of users. Indeed, media service providers are increasingly interested in evaluating the performance of their services provided as perceived by end users, in order to improve and better understand the needs of their customers. Network operators are also interested in this measure to optimize network resources and even (re)configure network settings to increase user satisfaction. There are several ways to get information about perceived quality. On the one hand, there are subjective evaluations done in fully equipped laboratories to investigate the perception of the end user. On the other hand, there are objective quality measures [41], which are often used to study the measurable parameters of the entire system, describing the quality of service (QoS) in a technical way. However, these parameters cannot describe all the variables that influence the perception of quality on the end-user’s side. For this reason, a new measure, called “Quality of Experience (QoE),” has been defined to reflect the quality perceived by end users.

### 5. How modeling QoS/QoE in related work?

To evaluate QoS/QoE, we choose in this article to focus on video streaming services [42–44] and all related requirements for serving this type of application. We define the concept, metrics and KPIs used to model it before giving mathematical models helping to model it using real traffic.
5.1 Video streaming overview

Video streaming playback, real-time video chats applications, online TV have some specific conceptions as media stream is done over IP; Video can be live (rewound/paused/recorded) or on demand (sought/played/paused). This can be done on specific components such as Distributed File Storage, Edge Servers, DRM System and Client Player. As an overview of video streaming techniques and protocols, the most famous are Akamai, Google Global Cache, Limelight Networks where video content is replicated from the original server to several called replica located near end-users especially when we use protocols based on HTTP. At the end, HLS or MPEGDASH known as delivery protocols are supported by video playback devices.

To distribute video on the internet, we find two video delivery methods: progressive download (process based on TCP/FTP or TCP/HTTP) or HTTP Adaptive Streaming (in order to solve video downloading problem). Streaming techniques from literature are described below:

- Progressive Download: where a client have the choice to get the appropriate codecs to play data using HTTP or HTTPS.
- HTTP Live Streaming: MPEG family transport channels offering live and on demand media.

In Transport layer, TCP is the most used protocol that offers “guaranteed delivery.” The second one is the unreliable UDP used in real-time operation. However, thanks to bandwidth availability, video is now delivered very efficiently using TCP caches placed the nearest to the client equipment. In Application layer, the most known protocols are HTTP (text/binary data) and RTP (audio/video over IP networks and streaming media).

5.2 Modeling QoE in a video streaming context

As we have discuss, there have been a number of research works done with the objective of video traffic QoE estimation based on media parameters (PSNR, VQM, SSIM, and PEVQ). However, very limited works can be found on QoE estimation of video transmission (Table 5). For other multimedia application over the internet, Various QoE models are developed for specific applications such as web browsing, audio/video services, online gaming and telepresence [45].

In order to evaluate QoE, we can use some QoS parameters such as packet loss, burst loss, jitter, delay, GoP-Group of Picture length, summarized in Table 4 as QoV, QoA and QoS indicators in Table 6.
5.3 Mathematical background

Nowadays, multimedia services of audio, image, video or data traffic know a fast progress on both public and private networks. The relationship between QoS and QoE relationship is not linear so it difficult to be estimated. Moreover various QoS/QoE algorithms can be found; Table 7 summarizes the mathematical methods that model QoE.

<table>
<thead>
<tr>
<th>Table 5 Application.</th>
<th>Impact of:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web browsing</td>
<td>Session time and temporal correlations</td>
</tr>
<tr>
<td>Audio services</td>
<td>PESQ (Perceptual Evaluation of Speech Quality) delay, echo</td>
</tr>
<tr>
<td>Video services</td>
<td>PSNR, MSE: codec type, video frame and packet loss</td>
</tr>
<tr>
<td>Online gaming</td>
<td>Quality factors such as interactivity and consistency</td>
</tr>
<tr>
<td>Telepresence</td>
<td>Interactivity, consistency and vividness</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 6 Indicators.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality of video indicators (QoV)</td>
</tr>
<tr>
<td>Quality of Application indicators (QoA)</td>
</tr>
<tr>
<td>Quality of Service indicators (QoS)</td>
</tr>
</tbody>
</table>

5.3 Mathematical background

Nowadays, multimedia services of audio, image, video or data traffic knows a fast progress on both public and private networks. The relationship between QoS and QoE relationship is not linear so it difficult to be estimated. Moreover various QoS/QoE algorithms can be found; Table 7 summarizes the mathematical methods that model QoE.

6. Video streaming protocols

Despite scarce resources available, mobile operators are facing the challenge to serve multiple clients while maximizing their QoE. New techniques, called adaptive streaming, are introduced for video transmission over varying channels such as wireless network. In order to match the available channel bandwidth, these techniques vary continuously both transmitted video quality and bitrate. To adaptively select video rates, video clients use those video streaming protocols. Video rate adaptation approaches use two factors to maximize viewer QoE: video quality or video rates and
<table>
<thead>
<tr>
<th>Method</th>
<th>Work done with it</th>
<th>How it is working?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fuzzy logic</td>
<td>Neural network and fuzzy logic formulae [46]</td>
<td>Prediction of the user satisfaction</td>
</tr>
<tr>
<td>Rough set theory</td>
<td>Inference rules for fuzzy expert system [47,48]</td>
<td>Subjective data set in the form of a conditional attribute set and a decision attribute set</td>
</tr>
<tr>
<td>Fuzzy expert systems [49]</td>
<td></td>
<td>Making decision with imprecise information</td>
</tr>
<tr>
<td>Regression analysis</td>
<td>Web service QoE estimation method [50]</td>
<td>Calculates indexes of a correlation function from the subjective test data; however, the quality estimated by this method has high MOS error margin</td>
</tr>
<tr>
<td>Artificial intelligence</td>
<td>Artificial Neural Networks (ANN) [51]</td>
<td>Connects the QoE directly to QoS metrics according to the corresponding level of QoE which was estimated by employing a Multilayer Artificial Neural Network (ANN)</td>
</tr>
<tr>
<td>techniques</td>
<td>Random Neural Networks (RNN)[52]</td>
<td>Adjusts the input network parameters in order to get the ideal output and satisfy QoE based on those parameters the source Bit Rate (BR), the Frame Rate (FR), the Packet Loss Rate (PLR), the Consecutively Lost Packet (CLP), and the Ratio of the encoded (RA)</td>
</tr>
<tr>
<td></td>
<td>Adaptive Neural Inference System ANFIS [53]</td>
<td>Learning technique can be found in the work of Khan [39] which presented new models to estimate the video quality based on an Adaptive Neural Inference System (ANFIS) Work was limited to four parameters: FR, SBR, packet error rate (PER), bandwidth</td>
</tr>
<tr>
<td></td>
<td>Machine Learning [19]</td>
<td>Decision trees (DTs), random neural networks (RNNs), hidden Markov models (HMMs), Bayesian networks (BNs) and dynamic Bayesian networks (DBNs)</td>
</tr>
<tr>
<td>Learning logic networks</td>
<td>[54]</td>
<td>Evaluating the QoE of web services</td>
</tr>
<tr>
<td>Neural networks</td>
<td>PSQA metric [55]</td>
<td>Numerical values such as packet loss percentage and mean loss burst size to output MOS based on RNNs</td>
</tr>
</tbody>
</table>
rebuffering time. There are three mainstream video rate adaptation methods: Network capacity based method, Buffer state based method and Hybrid method based on (1) HTTP caches placed in the network in order to reduce server load, (2) video split in smaller segments called chunks, (3) different video representation to permit users to request the one that matches at best their capabilities. We can give here some details about main streaming protocols.

Table 8 gives QoE metrics and Table 9 gives a survey of streaming protocols used and how they work.

At the end of this survey, we can argue that many multimedia streaming applications are still using HTTP. Using HTTP has some disadvantages: this protocol is at first designed for best effort and not for real-time multimedia transport, and second, it is unable to follow bandwidth variations of IP-based networks. For traditional RTP/UDP streaming, adaptation is difficult, some new studies propose adaptive HTTP streaming to encode content at multiple bit-rates. In wireless context, authors of Ref. [71] propose a schema for multi-user DASH that optimizes the media delivery to multiple clients.

Over the past years, authors propose methods for video quality assessment. Most of them [72,73] are measuring a visual quality of video frame called video spatial quality and withdraw the temporal effects. In streaming session [74], some new models are proposed for video temporal quality, but they don’t include the variation of bit rate (visual quality). In addition, Jiang et al. [75] proposed FESTIVE a general bitrate adaptation framework that implies a number of methods that fight to achieve a trade-off between video stability, fairness and efficiency. They used a dynamic rate selection heuristic is used to compensate for the biased interaction between the bit rate and the estimated bandwidth, as well as to make a compromise between stability and efficiency using a delayed update approach. But the problem of this model is the instability with the increase of the number of clients in the network and the effect of the bandwidth overestimation. In addition, it is not very sensitive to bandwidth fluctuations, which can lead to a significant undershoot of the buffer. It struggles to ensure fair bandwidth between competing DASH clients in a shared network environment. Tasnim et al. [76] proposed a novel approach based on Machine Learning and adaptive bitrate to enhance the video quality perceived and ameliorate the QoE in SDN context, they used Machine Learning to estimate the best level of the following video segment based on the current network conditions and client requirements (device, video content requested, etc.), in a second step authors proposed an algorithm to select the suitable quality level depending on the

Asma Ben Letaifa
<table>
<thead>
<tr>
<th>Internet traffic service</th>
<th>VoIP [56]</th>
<th>Video [57]</th>
<th>Video streaming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specificity</td>
<td>Streaming protocols control the data transfer between the multimedia server and the clients</td>
<td>Comparing performance of various Voice codecs, such as G.711, GSM, iLBC, Speex, and Skype’s codec</td>
<td>YouTube monitoring evaluate QoE of MP4, Flash Video (FLV) as well as WebM video in Standard Definition (SD) and High Definition (HD) format</td>
</tr>
<tr>
<td>Based on</td>
<td>RTP</td>
<td>PSNR (Obj) evaluation of the quality difference among pictures</td>
<td>HTTP bases streaming uses TCP for reliable data delivery</td>
</tr>
<tr>
<td></td>
<td>RTP over UDP for real-time transfers</td>
<td>VQM: perceptual effects of video impairments including blurring, unnatural motion, global noise, block distortion color distortion, and combines them into a single metric</td>
<td>[61] Client based video quality estimation approach, Passive YouTube QoE Monitoring for ISPs approach, Network Monitoring in EPC system</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MPQM: incorporates two human vision characteristics: contrast sensitivity and masking</td>
<td>QoE indicators of HTTP video streaming: Number of stalling events and their respective lengths, Initial buffering time: denoting the time to fill up the application buffer and to start the playback</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SSIM: is using the structural distortion measurement instead of the error</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RMSE: average squared error</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NQM: a degraded image that has been subjective to linear frequency distortion and additive noise injection</td>
<td></td>
</tr>
</tbody>
</table>

Continued
Table 8 QoE metrics in multimedia applications.—cont’d

<table>
<thead>
<tr>
<th>Services</th>
<th>VoIP [56]</th>
<th>Video [57]</th>
<th>Video streaming</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Internet traffic service</strong></td>
<td>IPTV–VOD</td>
<td>Skype–Viber</td>
<td>Video image</td>
</tr>
<tr>
<td><strong>Voice over Internet Protocol (VoIP)</strong></td>
<td></td>
<td></td>
<td>Youtube HTTP-based streaming</td>
</tr>
<tr>
<td><strong>Advantages</strong></td>
<td>Small delay</td>
<td>VQM: objective video quality metric, good correlation with human perception</td>
<td>Neither video nor audio suffer from impairments caused by lost data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MPQM: giving good correlation with subjective tests for some material</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RMSE and SSIM are the metrics that measure the error or similarity between a referenced image and a fused image</td>
<td></td>
</tr>
<tr>
<td><strong>Disadvantages</strong></td>
<td>UDP unreliable protocol</td>
<td>VQM doesn’t consider the impairment due to level variation</td>
<td>Rebuffering events alter the temporal structure of the video and result in stalling events so degrade QoE</td>
</tr>
<tr>
<td></td>
<td>Packet loss in multimedia streams Distortion of content</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Protocols</td>
<td>Definition</td>
<td>How does it work?</td>
<td></td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>HTTP Live streaming</td>
<td>Created by Apple to communicate with iOS and Apple TV devices and Macs running OSX</td>
<td>HLS can distribute both live and on-demand files</td>
<td></td>
</tr>
<tr>
<td>protocol [62]</td>
<td></td>
<td>Uniq technology available for adaptively streaming to Apple devices</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>HDS provides tools for integration of content preparation into existing encoding workflows</td>
<td></td>
</tr>
<tr>
<td>Adobe’s HTTP Dynamic</td>
<td>HDS enables on-demand and live adaptive bitrates video delivery of standards-based MP4 media over regular HTTP connections</td>
<td>Consumers get minimal buffering and fast start-up time, by adapting the quality of the video stream in real-time based upon the consumer’s changing bandwidth and CPU conditions</td>
<td></td>
</tr>
<tr>
<td>Streaming</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Microsoft’s Silverlight</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Smooth Streaming</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTTP Adaptive Streaming</td>
<td>HAS: a new way to adapt the video streaming quality based on the observed transport quality</td>
<td>The concept is based on the idea to adapt the bandwidth required by the video stream to the throughput available on the network path from the stream source to the client The adaptation is performed by varying the quality of the streamed video and thus its bit-rate, that is, the number of bits required to encode 1 s of playback. The aim is to divide the video stream into segments and to encode each of the segments in multiple quality levels, called representations. Based on his estimation of the available throughput, a client might request subsequent segments at different quality levels in order to cope with varying network conditions</td>
<td></td>
</tr>
<tr>
<td>Services [30,31,63–65]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jarnikov et al. [66]</td>
<td>Proposed to calculate the adaptation strategy using a Markov decision process</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Akhshabi et al. [67]</td>
<td>Evaluated two major commercial players and one open source player that use the technology of adaptive streaming over HTTP</td>
<td>Three aspects: reaction to persistent or short-term throughput changes, ability of two players to properly operate on a shared network path, and if the player is able to sustain a short playback delay and thus perform well with live content</td>
<td></td>
</tr>
</tbody>
</table>

Continued
<table>
<thead>
<tr>
<th>Protocols</th>
<th>Definition</th>
<th>How does it work?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zink et al. [68]</td>
<td>Investigate the impacts of video quality variations on human perception</td>
<td></td>
</tr>
<tr>
<td>The authors of Ref. [69]</td>
<td>Implemented an algorithm which aims at avoiding interruptions of playback, maximizing video quality, minimizing the number of video quality shifts and minimizing the delay between user’s request and the start of the playback</td>
<td></td>
</tr>
<tr>
<td>DASH: Dynamic Adaptive Streaming HTTP</td>
<td>Based on 3GPP’s AHS and the Open TV Forum’s HAS. It specifies use of either fMP4 or transport stream (TS) chunks or an XML manifest. DASH is a worldwide standard for adaptive streaming of video, audio and other media such as closed captioning.</td>
<td>DASH enables delivering media content from conventional HTTP web servers. DASH works by splitting the media content into a sequence of small segments, encoding each segment into several versions with different bit rates and quality, and streaming the segments according to the requests from mobile client. On the mobile device side, the DASH client will keep monitoring the network and dynamically select the suitable version for the next segment that need to be downloaded, depending on the current network conditions. Adaptive Video Streaming Measurements: Largescale studies of DASH performance have been conducted in commercial video streaming platforms such as Hulu, Netix and Vudu [31]. Similar measurement studies also covered aspects of the transmission behavior of video clients [67], DASH network traffic characteristics [70], and DASH QoE [45]. [71] proposes a QoE-driven multi-user DASH scheme that optimizes the adaptive HTTP media delivery to multiple clients in a wireless cell.</td>
</tr>
<tr>
<td>SCALABLE VIDEO CODING (SVC)</td>
<td>Least Recently used (LRU), Least Frequently used (LFU) or Chunk-based Caching (CC) are some different cache replacement algorithms [70]</td>
<td>Improves the cache performance based on some especial criteria/metric.</td>
</tr>
</tbody>
</table>
estimated best level, the choice is based on the buffer occupancy and the maximum number of intermediate quality video segments to be downloaded. Besides, authors in Ref. [20] presented an SDN-assisted QoE Fairness Framework, called QFF, where they try to enhance the heterogeneous DASH multiuser QoE by taking into account two main constraints, namely, the devices’ resolutions and current network requirements. This framework is based on SDN network and the OpenFlow protocol. However, this model neglects the buffer occupancy metric that’s why it can’t support a large number of clients. In another work [77] authors proposed the Server and Network-assisted DASH (SAND) architecture where they are based on the fact that client-driven rate adaptation provides less control for the network and service providers. This model introduces a centralized control component that offers asynchronous network-to-client and network-to-network communication. It receives QoE metrics from the clients and returns network feedback measurements, which are used by clients heuristically in their adaptation logics. SDNDASH proposed by Abdelhak et al. [22] aims to ease the scalability issues and improve the QoE. This architecture manages and allocates the network resources dynamically for each client based on its expected QoE by taking into account different metrics that affect the QoE (e.g., buffer sizes, display resolutions, quality requirements, available bandwidth, latency, etc.). Experimental results show that the proposed architecture significantly enhances scalability by improving per-client QoE by at least 30% and supporting up to 80% more clients with the same QoE compared to the conventional schemes.

Authors of Refs. [78,79] considered both spatial and temporal quality, and studied the impact of bit rate variation on user experience and developed a quantifiable measure DASH video. We can conclude with an assumption for DASH video, that user experience mainly depends on both subjective (temporal quality-initial delay and spatial quality-level variation) and objective factors (initial delay, stall or frame freezing and level variation).

7. Enhancing QoE approaches

7.1 Machine learning approach

Machine learning (ML) can be defined in Ref. [80] as “A study of making machines acquire new knowledge, new skills and reorganize existing knowledge.” ML is used in three approaches: supervised, unsupervised and semi-supervised. Four basic types of learning are known: (1) Classification (or supervised learning), (2) Clustering (or unsupervised learning), (3) Association, (4) Numeric prediction. ML algorithms (Table 10) can help in
<table>
<thead>
<tr>
<th>ML algorithms</th>
<th>How they work</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Resulting decision tree</strong></td>
<td><strong>k-nearest neighbors (kNN) algorithm</strong>&lt;br&gt;The k-nearest-neighbor algorithm is a pattern recognition model that can be used for classification and regression.&lt;br&gt;Often abbreviated to k-NN, the k is a positive integer, typically small. In classification or regression, the entry will consist of the k closest training examples in a space. In our work we focus on the k-NN classification. In this method, the output is class membership. This will assign a new object to the most common class among its k nearest neighbors. In the case of ( k = 1 ), the object is assigned to the class of the nearest neighbor.</td>
</tr>
<tr>
<td><strong>Naïve Bayes (NB)</strong></td>
<td>Selects optimal (probabilistic) estimation of precision values based on analysis of training data using Bayes’ theorem, assuming highly independent relationship between features</td>
</tr>
<tr>
<td><strong>Best-first decision tree</strong></td>
<td><strong>(BFTree)</strong>&lt;br&gt;Uses binary splitting for nominal as well as numeric attributes and uses a top-down decision tree derivation approach such that the best split is added at each step.</td>
</tr>
<tr>
<td><strong>Regression tree</strong></td>
<td><strong>representative (REPTree)</strong>&lt;br&gt;For general use, decision trees are used to visually represent decisions and show or inform decision making. When working with machine learning and data mining, decision trees are used as a predictive model. These models map the data observations and draw conclusions about the target value of the data. The goal of decision tree is to create a model that will predict the value of a target based on input variables. In the predictive model, the attributes of the data that are determined by the observation are represented by the branches, while the conclusions about the target value of the data are represented in the leaf. When learning a tree, the source data is divided into subsets based on an attribute value test, which is repeated recursively on each of the derived subsets. Once the subset of a node has the value equivalent to its target value, the recursion process will be complete. A fast implementation of decision tree learning which builds a decision/regression tree using information gain and variance with reduced-error pruning along with backfitting.</td>
</tr>
<tr>
<td><strong>Decision tables and naïve Bayes (DTNB)</strong></td>
<td>An hybrid classifier which combines decision tables along with naïve Bayes and evaluates the benefit of dividing available features into disjoint sets to be used by each algorithm, respectively.</td>
</tr>
<tr>
<td><strong>Bayesian network</strong> (BayesNet)</td>
<td>An acyclic directed graph that represents a set of features as its vertices and the probabilistic relationship among features as graph edges. It is employed for different types of Internet traffic including peer-to-peer (P2P) and content delivery traffic, and also for Internet traffic flow classification at a high speed.</td>
</tr>
</tbody>
</table>
monitoring networks or managing QoE like it is described in Table 6. They treat estimation [81] or cross-validation [82] or Simple regression models. For video applications, QoE prediction can be performed by applying mathematical models based on QoS parameters [83], full-reference algorithms VQM [84], WFL based model [85]. Paper [86] shows the relationship between QoS (traffic characteristics) and QoE (linear, logarithmic or exponential). To evaluate the QoE, studies based on user engagement, (resilience) abandonment rate, and frequency of visits are described in Ref. [87] and are using statistical tests (e.g., Pearson, Kendall). At the end, let’s describe an interesting ML_QoE algorithm which employs supervised regression, in which the predictors are metrics (such as jitter, packet loss, rebuffering, startup delay, resolution) and the predicted outcome is the QoE score.

### 7.2 Adaptive coding

With the variation in the number of active users in SDN networks, we need sometimes to select the best codec and adapting it to delivered data. Voice over Internet Protocol (VoIP) is considered as much used protocol for voice service delivery. A fixed voice codec between a VoIP server and clients without consideration of the network conditions will cause inefficiency. To improve the VoIP quality, three factors should be considered: the optimized codec selection at first, second, VoIP flow control, and finally the consideration of network congestion. The authors of Ref. [88] propose the adaptive Mobile VoIP (mVoIP) service architecture based on network QoS parameters and predicted mVoIP QoE.

### 7.3 Flow classification and monitoring QoE

With increasing number of users and items, the network operators and service providers need methods to monitor the quality of the video services. The monitoring and prediction should be performed in real-time and in different parts of the network where subjective assessments are very hard to perform but objective measurement methods are done. New methods are required to help calculating QoE Key Performance Indicator (KPI) from QoS parameters. QoE monitoring fields know especially those approaches: flow classification, flow monitoring, location aware monitoring, and collaborative filtering. Flow Monitoring represents the method to determine the number and duration of re-buffering without access to user terminal.
7.4 Buffering impact

Network buffering takes place in hosts, switches and routers, impacts network performance and contribute to delays, jitter, and packet losses. The author of Ref. [89] presents a study on the impact of buffer sizes on QoE, in such applications: voice, video, and web browsing. The author provides also insights into the extension of QoE metrics for client-side error recovery. A hit rate analysis of caching schemes is performed to optimize Web QoE by focusing on YouTube video popularities.

7.5 Path optimization

In SDN, service providers have a complete control on their servers, networks and video players but no control done on access networks. Several researches deal with this problem. Zafar et al. [25] use a ML based traffic classification technique to identify path for each application types in SDN. Ali et al. [26] introduced MPLS-TE traffic engineering with OpenFlow. Saurav et al. [27] studied traffic aggregation in a packet circuit network. In the same context, authors of Ref. [90] propose to monitor streaming flow in real time and dynamically change routing paths using MPLS. They measure various video QoE metrics at video players running on user terminal in order to dynamically change routing paths.

This exhaustive state of the art about protocols and used techniques to enhance QoE in video streaming services lead us to propose an appropriate methods based on SSIM metric and MI algorithms. Section 8 describes proposed method.

8. Simulation environment and testbed

Before presenting our approach based on SSIM and Machine learning algorithm over SDN networks, we have to put the environment, to describe topologies, scenarios and metrics to capture in order to best illustrate our approach. So, for this part of our paper, we choose to emulate some scenarios on SDN tools (Mininet and Open vSwitch) and experiment some politics in such environment. We will test some topologies making a variation in controller number, putting a Video streaming traffic with VLC server and measuring QoS parameters such as Jitter, RTT and bandwidth. The tests will be done on 2-min videos and ask users with web application to make a vote on those video qualities and generate a summary for this vote. Our politic to optimize end user quality of experience is to adapt a proposed flow to controller condition and topologies.
8.1 Testing on Mininet

For this part, we use Mininet and floodlight as an SDN simulation environment. We will make different topologies to evaluate the performance. So, in this section, we are going to measure three performance keys: Round-Trip Time, Jitter and Packet loss. Those measurements will be done on different topologies based on one Floodlight controller, two hosts and for four, six and eight switches for each case using UDP and TCP protocols. First of all, RTT (round-trip time) allows operators to understand the performance and help optimize their environment. Second, Jitter in IP networks represents the variation in the latency on a packet flow. Jitter is caused by congestion or path changes. It causes a hard problem in real-time communications. Finally, Packet loss is done when packets don’t arrive to destination. This event can cause noticeable effects in all types of digital communications. In this section, we tried to make simple topologies composed of 4, 6 or 8 switches \([s1, \ldots, s4], (s1, \ldots, s6), (s1, \ldots, s8)\), 2 hosts \((h1, h2)\) and 1 controller \((c0)\).

8.2 Graphical analyze

*RTT*: The round trip time has been increased when I increased the number of switches. Average 1 = 13,665 ms < Average 2 = 14,229 ms < Average 3 = 15,398 ms.

*Error rate*: The average in the three experiences is 0.25% which is a very low average. It’s true that when we increased the number of switches, the average has increased but it still has no influence in the transmission of UDP packets. The main reason of this loss is congestion. In fact, the UDP packet generator Iperf generates 10 Mbps which will be more than that by using the overhead of transmission (it is mainly equal to 11.9 Mbps after adding overheads). However, canal bandwidth is equal to 10 Mbps. These two facts generate the loss in the first switch. In conclusion, Mininet shows instability in its loss rate results but despite of that the loss rate is so low so Mininet provides as a good solution from the point of loss rate.

*Jitter*: Jitter also increased a little bit when the number of switches increased but it still low as a UDP packet transfer. Mininet didn’t provide us with reasonable results. We have to use another platform in order to have stable variation in any topology, while implementing video stream service.

We put in the table below the performance of our simulated network for a 10MB traffic, for next simulations, we attach a VLC server to the SDN controller and plan a 2-min video sequences and ask clients to vote for received quality in such networks. The testing is done on Mininet but if we had do it...
in real-time network, we already took at different times of the day because this variation is crucial to control for the variability in the bandwidth which is affected by rush hours in real networks. We have chosen a 2-min duration video to ensure that the viewer does not get bored and leave the test. The chosen video also had clear visual properties and wide homogeneous textures, so that we can easily detect the variation of perception as a function of the quality’s variation (Fig. 3).

8.3 Simulation scenarios

We run our three scenarios (Fig. 4) to make variation on Video Quality Score Initial Buffering Time and finally Buffering Frequency and measure each time the MOS obtained score.

Scenario 1: Mean Opinion Score as a function of Video Quality Score.

The VQS axis ranges from the first measured value which is about 29% to 95%. The value 29% is obtained from the lowest video quality the user might experience. The highest value on the scale is registered at 95% which can be translated in the fact that the user is watching the video at the maximum capacity of his terminal. The curve’s shape shows a diminishing return on VQS. In other words, for values of VQS between 29% and 50%, the same improvement in VQS yields to a higher increase in opinion score compared to the region between 50% and 95%. In fact, users who are used to a poor VQS are likely to appreciate a small increase in its value which consequently leads to a higher rating. This is also translated on the portion when values are
sup than 50% and above since the curve stabilizes and shows slight to no improvement in quality rating as a result of VQS increase.

Scenario 2: Mean Opinion Score as a function of Initial Buffering Time.

The initial buffering time takes values from 1s up to 29% s. The curve shows a heavy concentration around the interval between the values of 1 and 5 s. This can be explained by the strategy followed by the video provider which prioritizes the minimization of the IBT. The curve can be described in two separate regions. In fact, over the first region (1–11 s) users’ sensitivity to increasing IBT is not significant which can be translated in users’ tolerance to a maximum waiting time of 11 s. However, on the second region (13–29 s) the opinion score curve decreases significantly as IBT increases; the user’s experience becomes more and more unpleasant due to a higher waiting time.

Scenario 3: Mean Opinion Score as a function of Buffering Frequency.

The curve exhibits a downward sloping linear relationship between Opinion Score and Buffering Frequency per 8 s. The more stalls the viewer experiences while watching a video, the less immersion will be and the more annoying the experience gets.

8.4 Interpretation

The first and second curves taken together can infer to the fact that aiming for a minimum of 50% VQS and a maximum of 11 s in initial buffering time can maintain a high customer satisfaction. These numbers can be used as maximization and minimization constraints when optimizing the Quality of Experience. The third regression shows that buffering frequency is the one with the steepest linear relationship with the Opinion Score. This can be translated in the fact that no matter at what level of other KPIs the viewer is, any buffering is going to hurt his experience significantly. Therefore, buffering at the middle of the viewing experience is the factor hurting the experience the most for all users and should be given priority in optimizing the Quality of Experience.

9. Proposed approach: A ML-based enhancing QoE approach

9.1 SSIM approach

In this part, we investigate how Machine Learning approach works and how it can be used to improve quality of experience. We will start with an implemented work then we will expose our own solution. There are many Machine learning models of prediction. They aim to be able to predict the
QoE for end users without having to make subjective criteria for each new customer and media device. The work we will study involves the improvement of video quality experience through predicting SSIM measures: Structural Similarity. We expose the SSIM concept, some works from literature based on SSIM reference and Machine learning algorithms and finally, we let you discover our approach. SSIM is the measurement or prediction of image quality. It is based on an initial uncompressed or distortion-free image as reference. The idea of SSIM is to measure the structural similarity between the two images, rather than pixel to pixel difference. SSIM is an objective reference-based method to evaluate the quality of an image, which correlates well with the human perception and also allows an efficient computation. In the case of videos the SSIM is computed as the average SSIM over all the video frames. It is not easy to calculate the value of SSIM. So we will predict these measures through Machine Learning algorithms. Indeed, these algorithms are based primarily on a set of supervised learning techniques aimed at tackling discrimination and regression problems. First, unsupervised learning is used to extract an abstract representation of the raw data that capture the characteristics of the video. Then, supervised learning is performed to create a match between abstract representations and the corresponding SSIM coefficients of similar videos. Once the coefficients calculated, we determine the type of video encoding and the decision to take. So, in order to evaluate the objective QoE of a video, we will use the SSIM index, which is a complete reference measurement which measures the degradation of the image in terms of perception and of change in structural information.

SSIM is calculated by means of statistical parameters (mean, variance) calculated in a square window of size $N \times N$ (typically $8 \times 8$), which moves pixel by pixel in the entire image. The measures between the corresponding windows $X$ and $Y$ of the two images are calculated as follows:

$$SSIM(X, Y) = \frac{(2\mu_X \mu_Y + c_1)(2\sigma_{XY} + c_2)}{(\mu_X^2 + \mu_Y^2 + c_1)(\sigma_X^2 + \sigma_Y^2 + c_2)}$$

With $\mu$: Mean and $\sigma^2$: variance.

The SSIM index goes from 0 to 1, where 0 represents the extreme case of completely different and 1 the case of completely identical pictures. We can make also the correspondence between SSIM and the Mean Opinion Score (MOS), which evaluates the subjective video quality perceived on a scale of 5 values from 1 (poor) to 5 (excellent). So SSIM index 1 corresponds to excellent quality and therefore an index 5 scale MOS as shows Table 11.
Since the calculation of the SSIM can be difficult or impossible, we will use Machine Learning algorithms to predict the quality of the video that the user perceives subjectively. We will use then a data set of videos with their approximate SSIM index to teach the machine. So that it will be able to predict and calculate the quality of the video. Once the quality is calculated, we are going to use a set of optimization and operational research algorithms needed to choose the right encoding for the video dependent on its quality: this mechanism is called VAC: Video Admission Control. These algorithms have already been tested before. We have the possibility to double-check their performances. But, we will focus on studying the performance of our improved algorithm.

### 9.2 SSIM based works

With the rapid increase of streaming multimedia applications, there has been a strong and complex demand of Quality-of-Experience (QoE) measurement and QoE-monitoring technologies. Therefore, many techniques have been developed in order to evaluate as correctly as possible this perceptual quality. To inquire QoE measurement and predictions, most existing methods are based on the presentation quality of compressed video and objectives metrics such as PSNR, SSIM, MS-SSIM, etc., while others are interested on Machine Learning to predict the suitable user perception in real time based on previous measurements.

Rehman et al. [91,92] propose a full-reference video QoE measure, SSIMplus that provides real-time prediction of the video quality based on human visual system behaviors, display device properties (such as screen size, resolution, and brightness) video content characteristics, and viewing conditions (such as viewing distance and angle). We compared the performance of the proposed algorithm to the most popular and widely used full reference VQA measures that include Peak Signal-to-Noise Ratio (PSNR), Structural

<table>
<thead>
<tr>
<th>SSIM</th>
<th>MOS</th>
<th>Quality</th>
<th>Impairment</th>
</tr>
</thead>
<tbody>
<tr>
<td>≥0.99</td>
<td>5</td>
<td>Excellent</td>
<td>Imperceptible</td>
</tr>
<tr>
<td>[0.95, 0.99]</td>
<td>4</td>
<td>Good</td>
<td>Perceptible but not annoying</td>
</tr>
<tr>
<td>[0.88, 0.95]</td>
<td>3</td>
<td>Fair</td>
<td>Slightly annoying</td>
</tr>
<tr>
<td>[0.5, 0.88]</td>
<td>2</td>
<td>Poor</td>
<td>Annoying</td>
</tr>
<tr>
<td>&lt;0.5</td>
<td>1</td>
<td>Bad</td>
<td>Very annoying</td>
</tr>
</tbody>
</table>

**Table 11** MOS.

SSIM MOS Quality Impairment

---

Since the calculation of the SSIM can be difficult or impossible, we will use Machine Learning algorithms to predict the quality of the video that the user perceives subjectively. We will use then a data set of videos with their approximate SSIM index to teach the machine. So that it will be able to predict and calculate the quality of the video. Once the quality is calculated, we are going to use a set of optimization and operational research algorithms needed to choose the right encoding for the video dependent on its quality: this mechanism is called VAC: Video Admission Control. These algorithms have already been tested before. We have the possibility to double-check their performances. But, we will focus on studying the performance of our improved algorithm.

### 9.2 SSIM based works

With the rapid increase of streaming multimedia applications, there has been a strong and complex demand of Quality-of-Experience (QoE) measurement and QoE-monitoring technologies. Therefore, many techniques have been developed in order to evaluate as correctly as possible this perceptual quality. To inquire QoE measurement and predictions, most existing methods are based on the presentation quality of compressed video and objectives metrics such as PSNR, SSIM, MS-SSIM, etc., while others are interested on Machine Learning to predict the suitable user perception in real time based on previous measurements.

Rehman et al. [91,92] propose a full-reference video QoE measure, SSIMplus that provides real-time prediction of the video quality based on human visual system behaviors, display device properties (such as screen size, resolution, and brightness) video content characteristics, and viewing conditions (such as viewing distance and angle). We compared the performance of the proposed algorithm to the most popular and widely used full reference VQA measures that include Peak Signal-to-Noise Ratio (PSNR), Structural
Similarity (SSIM), Multiscale Structural Similarity (MS–SSIM), etc. They conclude that their algorithm is much better and faster tool than other traditional methods. Besides Lievens et al. [93] reported that the classical quality measurements (PSNR, SSIM, VQM) failed to predict the perceived video quality in HTTP adaptive streaming due to quality fluctuations and proposed an empirical quality metric to account for the streaming-specific distortions. Moreover, Thomas et al. [94] are based on the mapping method between SSIM, VQM and QoE to conduct an extensive measurement framework to study the influence of resolution of the video, scaling method, frame rate and video content types on the QoE. In other work [95] User experience is objectively expressed in terms of the average structural similarity (SSIM) index to analyze the video quality, authors propose a 4-degree polynomial approximation of the SSIM as a function of the coded video rate, they found that, measuring the QoE of a video in terms of SSIM, and expressing it as a function of the logarithm of the source rate, normalized to the full quality rate, it is possible to accurately approximate the quality vs rate distortion curve of the video as a 4-degree polynomial function. The polynomial coefficients can be tagged to the video and used for QoE-aware resource allocation. From their experiments they conclude that videos exhibit a relatively large SSIM (>0.94) even when the transmission rate is decreased to 1% of their full quality rate while others suffer a quick fall of SSIM already when the rate is reduced to 10% of the full rate. They conclude also that videos with similar SSIM attitude appear widely homogeneous in terms of scenes dynamics, with most dynamic videos exhibiting a quicker drop of SSIM as the RSF decreases. In Refs. [96,97] authors use some objective parameters such as SSIM, PSNR, NIQE, etc. to evaluate the influence of bitrates and stalling on the quality of experience, those metrics allow them to give an idea about the user’s opinion.

### 9.3 Machine learning based works

Other researchers use another method for estimating QoE for multimedia services based on Machine Learning algorithms, these algorithms give good results in real time; for example, Charonyktakis et al. [98] propose the MLQoE, a modular algorithm for user-centric QoE prediction. This framework employs multiple machine learning (ML) algorithms, namely, Artificial Neural Networks, Support Vector Regression machines, Decision Trees, and Gaussian Naive Bayes classifiers, and tunes their hyper-parameters for VoIP service. It uses also the Nested Cross Validation method to select the
best classifier and the corresponding best hyper-parameter values and predicts the performance of the final model. This model predicts accurately the QoE score. Specifically, a mean absolute error of <0.5 and median absolute error of <0.30 can be achieved. In other hand, Sajid Mushtaq et al. [99] have been achieved the correlation between QoS and QoE in search of capturing the degree of user opinion, they are based on Machine Learning to determine the most suitable one for the task of QoS/QoE correlation in order to study the effect of QoS metric on the QoE to deliver a better quality of service to end-users. This work evaluated six classifiers and determines the most suitable one for the task of QoS/QoE correlation. Experimental results show the in case of mean absolute error rate, it is observed that DT has a good performance as compared to all other algorithms. Also in Ref. [100], a machine learning technique is proposed using a subjective quality feedback. This technique is used to model dependencies of different QoS metrics related to network and application layer to the QoE of the network services and summarized as an accurate QoE prediction model. Finally, authors in Ref. [101] used Machine Learning for estimating audiovisual quality multimedia service. They used to train the models Random Forests and Multi-Layer Perceptron methods, results show that Random Forests based methods outperform Multi-Layer Perceptron methods in terms of RMSE, Pearson Correlation coefficient value and 95% confidence interval boundaries.

9.4 Proposed approach

We saw in the previous section, an example of an algorithm of Machine Learning to ameliorate QoE. Indeed, this algorithm when deciding at the end to choose the type of video encoding it only takes the quality expected by the user into account. But we can consider this approach as limited because we still have bandwidth problems, network congestion and other services that are running and that may affect the network quality and thus reduce the quality of experience and cause a cut during video streaming.

Now, we are going to detail our proposal of improvement to this algorithm to further enhance the user’s quality of experience. As we have already explained in the SSIM part: this solution is very interesting. Indeed, depending on the SSIM index, we will know the minimum required quality level and therefore the appropriate codec to encode the video and send it over the network.

But this choice of codec ignores the state of the network, limitations of bandwidth and anything can happen on the network. It only takes the
minimum quality of the video into account. But, sometimes this minimum is not even valid for the user’s network conditions and could provoke the cut of the video streaming which implies a very low QoE.

Our improvement addresses these network conditions. So, after calculating the quality of the video, we will not directly choose the codec to use. Using the SDN centralized architecture, we can predict the state of the network and know the different metrics that can influence on the transmission of the video and that can cause poor quality of experience. After calculating these measures, we will calculate the correlation between the SSIM index and network quality index. This will generate a new index from which we will choose the appropriate codec and so we will ensure a minimum QoE while taking into account two conditions: the state of the network and the transmission part as well as the minimum quality required for the video.

All this work is going to be in SDN architecture since we are going to control the network with the need of the controller. Then, as we have already explained, it is almost impossible to calculate the SSIM indicator, we can only predict thanks to Machine Learning algorithms, so we implemented these prediction algorithms and chose a free data set not very long videos (about 1 min and a half) to teach our machine to calculate the SSIM. Once calculated: this index gives us an idea about the minimum quality required for a video. All this work is done in the video streaming server which is responsible for the delivery of videos. The SSIM algorithm is already implemented and is open source so we downloaded it and we changed it. The algorithm in its original version calculates the similarity between two consecutive images. We will modify it to calculate an index for the entire video. Once the SSIM is calculated for the video, we need to calculate the state of the network to select the appropriate codec. It can be collected from the network using QoS metrics or predicted depending on the previous state. After that, we will calculate a new index that is the correlation between SSIM index and the state of the network. This index is the way to select the appropriate codec. Once the codec is selected and the video is being streaming and since that the network parameters may change and streaming video is a real-time service, the choice of codec will also be dynamic and can change from one moment to another.

We can see from Table 12 that each index is associated with a particular codec according to the algorithm already presented in the previous section. This codec will vary over time and will depend on network state in real-time.
9.5 Experiments

In this section, we will deploy in our test platform several scenarios and evaluate our solution’s performance. Our platform consists of two levels: First, the data plane that is the switches and the transmission part and then, the control plane, which consists of controllers and the commanding part. The QoE measure we will use for evaluating network performance is the Success rate: the percentage of packets received by the host compared to total packets exchanged. This measure will be then correlated with the SSIM index for selecting the best codec. To study the performance of this proposal, it is necessary to use different simulation scenarios. Indeed, we will test our platform for a real-time service that is video streaming.

9.5.1 First scenario

After deploying the entire architecture, now we’ll just start with a single user who will connect to our streaming video service, it will try to watch all the videos that exist in our Streaming video server without deploying our solution. It will therefore seek switches to reach the server. The number of nodes that we used in this first simulation is 10 and the user is only going to watch from the database of videos. We have 15 videos and we will restrict our study to the percentage of packets received successfully. In Table 13, we summarized the different results of the user’s access to our server for the 15 videos. We can notice that we have a situation here where the user is logged alone and therefore enjoys all of the bandwidth. So we have an average of 97% of success. This is due mainly to the fact that the user is alone and there is no congestion or other services launched at the same time to clutter nodes and decrease the quality perceived by the user.

9.5.2 Second scenario

In this second scenario, we will increase the number of nodes and the number of users requesting the same service at the same time and all trying to have streaming video service without implementing our idea and therefore this scenario will be seen as an extreme case scenario or saturation case. So

<table>
<thead>
<tr>
<th>Index</th>
<th>Chosen codec</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.66 → 0.99</td>
<td>x.264</td>
</tr>
<tr>
<td>0.33 → 0.66</td>
<td>H.264</td>
</tr>
<tr>
<td>0.00 → 0.33</td>
<td>MPEG4</td>
</tr>
<tr>
<td>Video</td>
<td>First scenario</td>
</tr>
<tr>
<td>---------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>Duration time</td>
</tr>
<tr>
<td>Video 1</td>
<td>1:35</td>
</tr>
<tr>
<td>Video 2</td>
<td>1:28</td>
</tr>
<tr>
<td>Video 3</td>
<td>1:06</td>
</tr>
<tr>
<td>Video 4</td>
<td>1:52</td>
</tr>
<tr>
<td>Video 5</td>
<td>1:26</td>
</tr>
<tr>
<td>Video 6</td>
<td>1:32</td>
</tr>
<tr>
<td>Video 7</td>
<td>1:41</td>
</tr>
<tr>
<td>Video 8</td>
<td>1:22</td>
</tr>
<tr>
<td>Video 9</td>
<td>1:17</td>
</tr>
<tr>
<td>Video 10</td>
<td>1:09</td>
</tr>
<tr>
<td>Video 11</td>
<td>1:27</td>
</tr>
<tr>
<td>Video 12</td>
<td>1:19</td>
</tr>
<tr>
<td>Video 13</td>
<td>1:41</td>
</tr>
<tr>
<td>Video 14</td>
<td>1:22</td>
</tr>
<tr>
<td>Video 15</td>
<td>1:52</td>
</tr>
</tbody>
</table>
we will use 40 switches and 10 users who all want to connect to the server at the same time and we will look to a single user and follow his trial to connect to the 15 videos.

In Table 13, we summarized the different results of the user’s access to our architecture for the 15 videos in this second scenario. Through the results, we can deduce that in a scenario of saturation or congestion when everyone wants to connect to the same server through the same resources, the quality perceived by the user decreases and this is confirmed by the average of 47% for the success rate of packets.

9.5.3 Third scenario
In the third scenario the same number of users trying to have that same service with a user who will try to watch the 15 streaming videos and nodes that provide this service implements the ameliorated SSIM solution and this will give us an idea on the behavior of our approach when there is a streaming service requested simultaneously for a normal user and a user implementing our solution.

In Table 13 we summarized the different results of the user’s access to our architecture for 15 videos after deploying our algorithm. We can easily notice how the percentage rose again to stabilize in the 83% rate of success of exchanged packets: this rate is better than the previous rate and in the same network conditions could lead us to conclude that our solution is efficient and improves the user’s QoE.

The graphic below shows the difference between the three scenarios (Fig. 5).

9.5.4 Results
We believe in the future we can test more factors and ameliorate our algorithm to consider single metric to be able to improve the quality of the experience. We also think of adding and improving our proposed solution so that it will have a better performance for other services such as video conference and real-time applications.
We will also look at other approaches. As we rather concentrated in this work on machine learning approach that leaves us even more prospects and opportunities to perform in the future to finally combine all solutions.

10. Conclusion, perspectives and futures work

Preliminary experimental results show that Mininet can effectively estimate the efficiency of adaptive solution and intelligent placed on controllers. This work allows claiming that our proposed approach is promising, as the simulation procedure for the QoE calculation can be efficiently performed. Further experimental research is needed to estimate the efficiency of this approach for different types of scenarios and their parameters. We argue that Mobile web platform is needed to allow the Internet service providers to collect these measures, analyze them and draw recommendations to improve the degree of their user’s satisfaction. In this work, we proposed and worked on a novel machine learning algorithm for objectively estimating QoE service based on SSIM parameters. Further simulations will be carried out on the suitability of both Mininet and Open vSwitch to calculate and predict QoE for video streaming services. In particular, this paper focuses on a QoE overview and discusses QoE-metrics and video streaming protocols. We propose an algorithm based on SSIM and Machine learning technique to enhance QoE in SDN networks. In the future, the performance of the proposed algorithm will be evaluated in real networks with more than one simulation environment.

References


network, in: Optical Fiber Communication Conference and Exposition (OFC/NFOEC) and the National Fiber Optic Engineers Conference, Los Angeles, USA, 2011.


[61] O. Hohlfeld, Impact of Buffering on Quality of Experience, Oliver Hohlfeld, 2013.


[70] iDASH—Yago Sánchez MMSYS-2011, ppt, fraunhofer.


[76] T. Abar, A. Ben Letaifa, S. Elasmi, Enhancing QoE Based On Machine Learning and DASH in SDN Networks, Accepted in WAINA18, Cracow, Poland, May 2018, IEEE.


About the author

**Dr. Asma Ben Letaifa** is an assistant professor and member of the MEDIATRON research lab at the Higher School of Communications, SUPCOM, University of Carthage, Tunisia. She holds a Telecom Engineering Degree from SUPCOM, University of Carthage, Tunisia and a PhD jointly from SUPCOM, University of Carthage and UBO, Université de Bretagne Occidentale, Brest, France.

Her research activities focus on telecom services, cloud and mobile cloud architectures, service orchestration, bigData and quality of experience in an SDN/NFV environment. She is author and coauthor of several articles on these subjects.

She is also author of several courses on telecommunications services, network modeling with queuing theory, web content, cloud architectures and virtualization, massive BigData content and machine learning algorithms. She is also coauthor of the “Linux practices” MOOC on the FUN platform.