Chapter 9
Evaluation and Design of Performable Distributed Systems

Naazira B. Bhat, Dulip Madurasinghe, Ilker Ozcelik, Richard R. Brooks, Ganesh Kumar Venayagamoorthy, and Anthony Skjellum

Abstract Performability measures system performance including quality, reliability, maintainability and availability over time, regardless of faults. This is challenging for distributed systems, since the internet was designed as a best-effort network that does not guarantee that data delivery meets a certain level of quality of service. In this chapter, we explain the design, test and performability evaluation of distributed systems by utilizing adversarial components. In our approach, the system design uses adversarial logic to make the system robust. In system test, we can leverage existing, powerful attacks to verify our design by using existing denial of service (DoS) attacks to stress the system.

Keywords Performability · Robust control · Game theory · Blockchain · Denial of service

9.1 Introduction

Critical infrastructure is geographically distributed and vulnerable. A performable system withstands and mitigates unfavourable conditions. Distributed system components share data and information. In practice, sensor malfunctions, data communication hijacking or external disturbances can occur. The system has to tolerate disturbances in operation. To design and test these systems, we integrate adversarial logic into our approach. Game theory is the mathematics used to model conflict between

N. B. Bhat · D. Madurasinghe · R. R. Brooks (✉) · G. K. Venayagamoorthy
Holcombe Department of Electrical and Computer Engineering,
Clemson University, Clemson, South Carolina, USA
e-mail: rrb@g.clemson.edu; rrb@acm.org

I. Ozcelik · A. Skjellum
SimCenter, University of Tennessee at Chattanooga, Chattanooga, Tennessee, USA

I. Ozcelik
Department of Computer Engineering, Recep Tayyip Erdogan University, Rize, Turkey

K. B. Misra (ed.), *Handbook of Advanced Performability Engineering*,
https://doi.org/10.1007/978-3-030-55732-4_9

rational decision-makers. In this chapter, we present a game theory-based approached for DCS design. Game analysis techniques improve performability.

We model the system as a Two-person Zero-sum (TPZS) game. We introduce different disturbances and countermeasures. We start by finding the best metric(s) to measure network performance. Zero sum means that the disturbances and countermeasures have no cooperation, the success of a disturbance is equal to the lack of performance of the countermeasure, and vice versa. Two person means we consider only affects on one system and cooperation between disturbance and countermeasure is impossible. If more than one metric is used, we either use a weighted sum of the individual metrics or we could always use the value of the worst metric (this is known as the H-infinity metric).

A game between these parties is established. For each component, we determine how it could malfunction to most severely sabotage the system. We then implement these behaviours. We then look at the other components to find how they could best counter these malfunctions to create a set of countermeasures.

We then create a set of simulations or experiments where we measure the results of each malfunction against each countermeasure. This provides a payoff matrix that we can use to analyze the different possible deviations from our initial design. The final resulting system should tolerate disturbances, which improves the performability of the system.

9.2 Game Theory-Based Robust Control Design

Game theory is the mathematics of competition scenarios. It models conflicts and cooperation between intelligent rational decision-makers. Game theory used general mathematical techniques to analyze the interactions in between parties that influence each other (Myerson [1]). It has been widely used in social science (Shubik [2]), economics (Leven [3], Chen [4]) and engineering (Trestian [5], Changwon [6]) applications. In our approach, we consider system deviations by positing that the system will fail in the worst possible way. We then embed in the system countermeasures that allow it to successfully adapt to these adverse conditions.

The rest of this section describes the mathematical tools we need to analyze the system. In Sect. 1.1.1, we describe the Z-test that tells us from our experimental analysis whether or not the effects of two specific malfunctions or countermeasures differ significantly. We then look in Sect. 1.1.2 as to whether one malfunction, or countermeasure, is always superior to another. If one approach is always superior (dominant), then we can discard the other one and simplify our design space. If we are lucky, the system recursively simplifies itself and we end up with a saddle point. For the saddle point, there is one failure mode that is more significant than all the others and it is tied to single countermeasure that best counters it. If this is not the case, Sect. 1.1.3, then the worst possible failure is a randomized combination of individual faults. Luckily, our approach finds this worst condition and also provides the best-randomized set of countermeasures to minimize the disturbance.

9.2.1 Z-Test

The Z-test is a statistical analysis based on the difference between the means of the test sample set and a mean of the population (Li [7]). This is a hypothesis test used for larger sample set (Bo [8]). Based on the central limit theorem, when the number of samples becomes larger, the sample average follows a Gaussian distribution with mean equal to the mean of the distribution and standard deviation, where σ is the standard deviation of the distribution and n is the number of the samples. The Z-test can be conducted on distributions, which Gaussian and the standard deviation are known (Hedge [9]). In our study, we use Z-test to compare two strategies in the game to find the dominant strategy under define significance level, which is useful in finding the saddle point of large payoff matrix. The selected level of significance (α) is 0.05. Considering two-tail tests, under the given significance level, we identify whether selected strategy (R) is dominant, is dominated or cannot conclude the dominance status compared with another strategy (S) based on the Z-score calculated using (9.1) and referring to Tables 9.1 and 9.2.

Table 9.1 Example game between players A & B

		Player B			
		J	K	L	M
Player A	P	$PO_{(PA)}$	$PO_{(PB)}$	$PO_{(PC)}$	$PO_{(PD)}$
	Q	$PO_{(QA)}$	$PO_{(QB)}$	$PO_{(QC)}$	$PO_{(QD)}$
	R	$PO_{(RA)}$	$PO_{(RB)}$	$PO_{(RC)}$	$PO_{(RD)}$
	S	$PO_{(SA)}$	$PO_{(SB)}$	$PO_{(SC)}$	$PO_{(SD)}$

Table 9.2 Dominance strategy selection conditions

Z-score	Conclusion
$(Z_{(R,S)} >= 1.96)$ for all Player B strategies	S is dominated by R
$(Z_{(R,S)} >= 1.96)$ for one or more Player B strategies & $(1.96 > (Z_{(R,S)} >= -1.96)$ for rest of the Player B strategies	S is dominated by R
$(1.96 > Z_{(R,S)} >= -1.96)$ for all Player B strategies	Uncertain
$Z_{(R,S)}$ values are on all three regions for all Player B strategies	Uncertain
$(-1.96 > Z_{(R,S)})$ for one or more Player B strategies & $(Z_{(R,S)} >= 1.96)$ for rest of the Player B strategies	Uncertain
$(-1.96 > Z_{(R,S)})$ for all Player B strategies	R is dominated by S
$(-1.96 > Z_{(R,S)})$ for one or more Player B strategies & $(1.96 > Z_{(R,S)} >= -1.96)$ for rest of the Player B strategies	R is dominated by S

$$Z_{(R,S)} = \frac{\bar{X}_R - \bar{X}_S}{\sqrt{\frac{\sigma^2_R}{n_R} - \sqrt{\frac{\sigma^2_S}{n_S}}}} \tag{9.1}$$

9.2.2 Dominant Strategies

Consider the game in Table 9.1. Consider two strategies *R* and *S* of Player A. Let us assume according to Table 9.2, the *R* strategy dominates the *S* strategy. We can remove the *S* strategy (a row in the example game) from the payoff matrix, since it is never a better choice than *R*. The same approach can be used for the Player B (column-wise).

9.2.3 Mixed Equilibria

John Forbes Nash Jr. won the Nobel Prize, by proving that every finite game must have at least one Nash equilibrium (Nash [10]). But there are instances where a game is not having a pure strategy Nash equilibrium. The matching penny game is an example. Hence, John Forbes Nash Jr. divides Nash equilibrium into two types, Pure Strategy Nash Equilibrium and Mixed Strategy Nash Equilibrium. To solve a mixed strategy and find Nash Equilibrium, we use a mixed strategy algorithm. Mixed strategy algorithm is based on probability distribution of pure strategies.

Let A choose option A and B with probability of p and $(1-p)$. Then, the following statements are derived and illustrated in Fig. 9.1's graph:

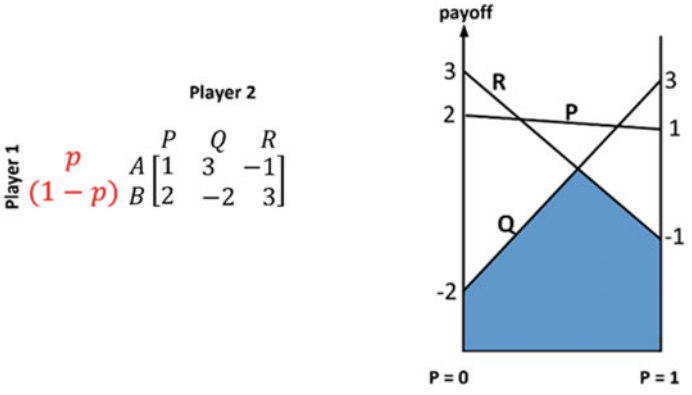

Fig. 9.1 Example game with mixed equilibrium

1.When player 2 chooses option P the payoff of player $1 = p + 2(1 - p)$.
2.When player 2 chooses option Q the payoff of player $1 = 5p - 2$.
3.When player 2 chooses option R the payoff of player $1 = 3 - 4p$.

A mixed equilibrium game is shown in Fig. 9.1. Player 1's optimal strategy can be found from the graph, which is to play option A with 0.5556 probability and option B with 0.4444 probability. The payoff of the game is 0.7778.

The procedure of the design approach process is shown in Fig. 9.2. This is the generic model that can be used for any distributed control system application.

9.3 Evaluating System Robustness

Once a distributed system has been designed to maintain performability, it becomes important to verify that the system is in fact successful. To do this, we consider common approaches for disabling or degrading distributed systems. Luckily, criminals have been attacking systems on the internet for decades. Over time, they have created a set of common attacks that are used to destroy internet applications. The most robust attacks can be classified as denials of service. Our performability verification leverages decades of work by these malicious actors to be certain that the system continues to provide service even under the harshest circumstances.

The most successful tools are generally referred to as Distributed Denial of Service (DDoS) attacks (Ozcelik [11]). To verify our systems, we therefore integrate successful DDoS methods into our verification suite. This section looks at system verification for distributed ledger technologies.

Blockchain distributed ledgers have become one of the most frequently considered solutions for ensuring security of the storage of data and its transfer through decentralized, peer-to-peer networks in recent years. Blockchains are a data structure based on shared, distributed and fault-tolerant database that every participant in the network can have access to, but none of those can tamper with it. Being a cryptographic-based distributed ledger, trusted transactions are enabled among untrusted participants in the network using the blockchain technology. Blockchains assume that malicious nodes are present in the participating network but rely on the computational capabilities of the honest nodes to ensure that the exchanged information is resilient to manipulation. The absence of a centralized entity speeds up the entire process. Owing to the cryptographic structure of the blockchain, it is challenging to alter it. Based on these features, blockchains have drawn attention from a wide range of stakeholders including academics, healthcare and other government agencies.

Blockchain has seen a surge in interest among researchers, software developers and other industry practitioners because of security features like immutability it offers (Kan [12], Miller [13], Fiaidhi [14], Samaniego [15]). However, an important aspect for which the blockchain-based networks need to be tested is their vulnerability to D/DoS attacks. If successful, the entire data stored on the application can be rendered

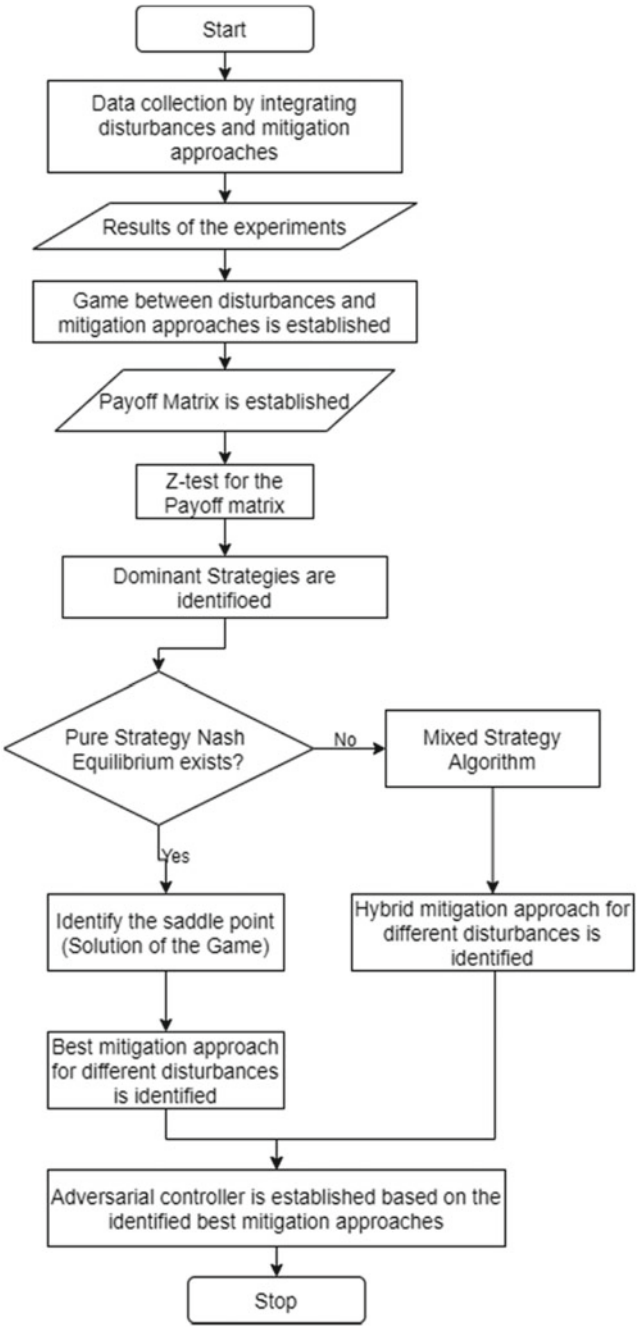

Fig. 9.2 Flow of the design process

useless and unavailable to its owners. This is even more critically important for time-critical applications such as healthcare. Therefore, network attacks on blockchain-based structures are to be taken seriously regardless of the intrinsic security properties of this data structure.

9.3.1 Case study—A Blockchain-Based (Distributed) System

The communication protocol used in our proposed distributed system is TCP. Therefore, we exploit the vulnerabilities of the TCP protocol for stress-testing our network. One of the most effective attacks against TCP is SYN floods. Hence, we focus on evaluating the performance of our proposed blockchain network in the presence of the SYN flood attack and its variants. It is described in Sect. 9.4.2 how the three-way TCP handshake is exploited to make the SYN floods effective. For our experiments, the attack is launched using IP-spoofing, randomized IP spoofing and the local area network denial-of-service (LAND) attack technique. The tool used to generate the attack traffic is Hping3, and the network analyzer used is Wireshark.

The tests have been conducted offline in the Network Security Lab at Clemson University's ECE department. The reason for offline testing is to ensure that the attack traffic does not disrupt the infrastructure of the university campus and that the effects of the attacks are contained within the lab.

Our distributed network uses Castro and Liskov's practical Byzantine-fault-tolerant (pBFT) algorithm to reach consensus. pBFT can tolerate a maximum value of faulty nodes (f) equal to less than $N/3$, where N is the total number of participating miners in each round. For our experiments, we used $N = 5$; and $f = 1$. The system architecture is shown in Fig. 9.3.

A brief description of the architecture components is as follows:

1. Clients:
   Clients submit data or transactions to the participating miners for registration over the blockchain. These are the users of our blockchain-based technology.

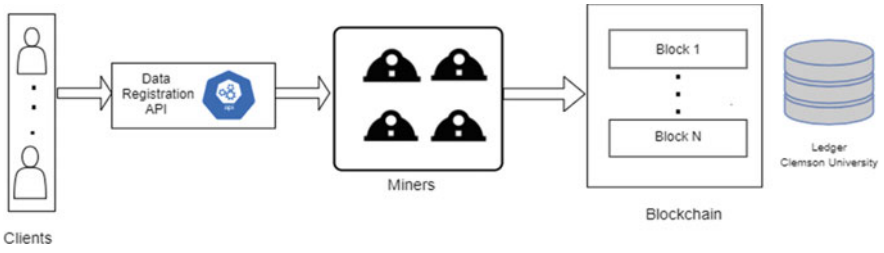

Fig. 9.3 Our distributed system architecture

2. Transactions:
   Transactions are the data or files stored on the blockchain. These are the backbone of the provenance. Transactions can reference previous transactions if they are not the first transaction themselves, or they can be genesis events, (i.e. the first data collected from a particular use case of our blockchain network). We store the SHA-3 hash of the transaction on the server instead of the transaction itself. This helps to drastically reduce the size of the blockchain.
3. Blocks:
   Blocks are one of the prime components of the system. A sequence of verified blocks forms the blockchain. The current block consists of hash of the previous block. This property makes the blockchain immutable. Blocks are added to the blockchain by miners or entities authorized to participate in the mining round.
4. Servers:

The servers are locally maintained and will hold the raw data comprising the ledgers in which the blockchains are held.

9.3.1.1 Experimental Setup

For the experiments, we have configured seven virtual machines (VMs) on seven different host machines—one on each host. A brief description of the components of the experiment is as follows:

1. Miners:
   There can be N participating miners. For our experiments, we use $N = 5$, four of which are honest, and one of which is
   malicious. Miners receive transactions submitted by the clients. Based on the algorithm described above, the selected miner mines the next block and appends it to the current length of the blockchain.
2. Client:
   Clients submit their files to the miners with the intention of making their data secure and immutable. For our experiment, we have initially configured a single client node, which is submitting xml files with a wait time of 3 s between each successive file submission to all the participating miners.
3. Attacking node:
   The attacking node can send attack traffic to any of the N participating miners. This node can be either a malicious miner participating in the mining process or a node external to the participating miners. For our experiments, the attacking node is one of the participating miners.
4. Hping3:
   This is the tool we use on the attacking node to generate attack traffic owing to its versatility and simple usage.
5. Wireshark:
   This is the tool we use as a network analyser on the victim's machine.

Blockchain update requests could be sent by either a new miner who attempts to join the mining network and needs to retrieve the length of blockchain mined so far before being able to participate in subsequent rounds or by a new client who joins using IP address of any miner but needs to know to the peer list of miners such that they are able to send transactions to all the participating miners simultaneously. For our experiments, we assume that the blockchain request is sent by a new miner who wants to join the mining network.

9.3.1.2 Results

To evaluate the performance of the blockchain-based network under the influence of each attack, we determine the amount of time the network takes to send a response to the blockchain update request by the requesting miner. This is the most strenuous step in the functioning of this distributed network. If the network is able to withstand torture testing in this step, we assume that it would perform well in rest of the steps. The miner who sends the response to the blockchain request is randomly selected and the selection is equiprobable. For reference, we also checked how much time it takes for the new miner to get the response when the network is not under the influence of any attack. It should be noted that SYN cookies are enabled on each of the machines involved in our experiments.

For all the cases, each of our mining rounds was successfully executed and the miner was randomly selected to produce the next block. The results for the response times are summarized in Table 9.3 and Fig. 9.4.

Table 9.3 Response times for different attack cases

Case	No. of attack packets sent		Response time of the victim [seconds]	
			When miner under attack gets selected to send the response to the blockchain request	When miner that is not under attack gets selected to send the response to the blockchain request
No attack	_		_	5
SYN flood with IP spoofing	Spoofed IP reachable	612195	20	5
	Spoofed IP unreachable	1643336	48	5
SYN flood with randomized IP spoofing		926612	30	5
LAND attack		1567141	47	5

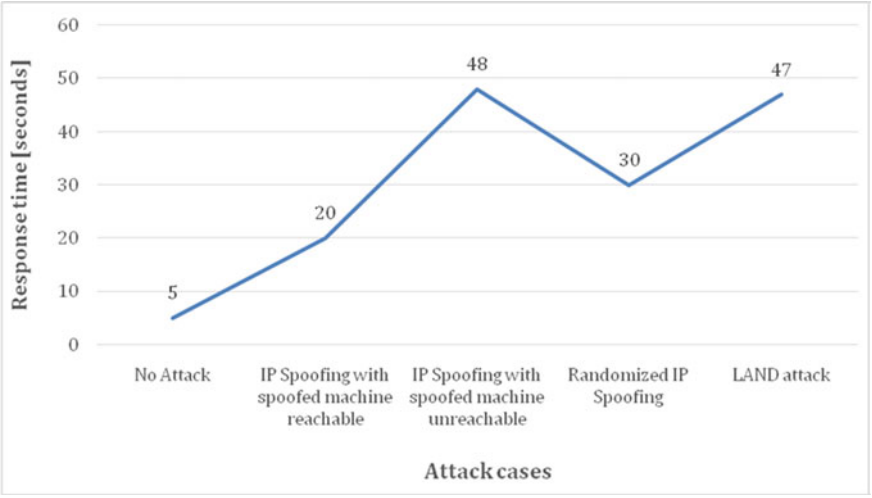

Fig. 9.4 Graphical representation of response times in case of no attack and different sub-cases of the SYN flood attack

9.3.1.3 Analysis

From results in Sect. 2.1.2, we observe that the network performance changes under the influence of an attack by a considerable extent. When the miner who gets selected to send the response to the blockchain request is under attack, the response time increases. The reason for the increase in response time is the incomplete TCP connection requests in the victim's TCP queue as a result of a large number of SYN packets sent by the attacker. On receiving the SYN packets, the victim responds with the SYN-ACK packet and keeps waiting for the final ACK packet from the node that initiated the SYN request. Until the TCP queue is reaped from the incomplete connection entries, the victim's resources get throttled. As a result, miner cannot process the requests coming from legitimate users while it is attacked. This is explained in detail in Sect. 9.3. The attack with maximum impact is the SYN flood using IP spoofing with spoofed machine not reachable. The maximum number of attack packets is sent in this case to the victim. The reason lies in the TCP parameter 'syn-ack-retries'. The victim is configured to keep retransmitting the SYN-ACK packet in response to the initial SYN request for a certain number of times. Since the attacker employed a spoofed IP address, and kept that machine powered off, the victim has to spend an additional time in trying to reach the machine first.

In all the attack cases with the victim being the selected miner to send the response, the performance of the network is degraded. This raises an important question about the reliability of distributed systems. Even though blockchain claims to ensure security and immutability, delay in the response of the network can cause many harmful implications. The stakes are higher for critical applications such as banking, healthcare and other important infrastructure.

However, an important observation is made when an unattacked miner gets selected to send the response to the blockchain request. The response time in this case is the same as when no attack is launched. This means potentially the attacker failed to launch a successful DoS attack on our network in this case. This implies that the probability of users experiencing denial-of-service while using our network is equal to the probability that an attacked miner gets selected to send a blockchain update response.

As mentioned earlier, the miner selection is random and equiprobable. Thus, the selection probability follows uniform distribution. So we can mathematically express this as:

$$P(\text{Successful DoS attack on system}) = P(\text{Selected min er to send blockchain updates be the victim})$$

$$P(\text{Selected miner to send blockchain updates to be the victim}) = 1/N$$

Therefore, for our experiments, where $N = 5$:

$$P(\text{Successful DoS attack on system}) = 1/5$$

As N increases, the probability of a successful DoS attack on our distributed system decreases for a constant number of miners under attack simultaneously.

9.3.1.4 Inference

This raises an important question about the reliability of distributed systems. Even though blockchains claim to ensure security and immutability, delay in the response of the network can cause many harmful implications. The stakes are higher for critical applications such as e-banking, healthcare and other important infrastructure. Thus, distributed systems such as blockchains do not necessarily offer a considerable amount of availability and reliability unless designed to perform robustly.

9.4 Denial of Service

A Denial of Service (DoS) attack intentionally disables a system or a service to its legitimate users. Criminals generally perform these attacks by targeting the limited resources of a system. If an attacker uses more than one node to perform these attacks, it is called Distributed Denial of Service (DDoS) attack.

Targeting scarce system and network resources are common approaches used to perform a denial of service attack. These attacks are called resource-starvation attacks. The attacker may target system resources such as memory, disk space, CPU

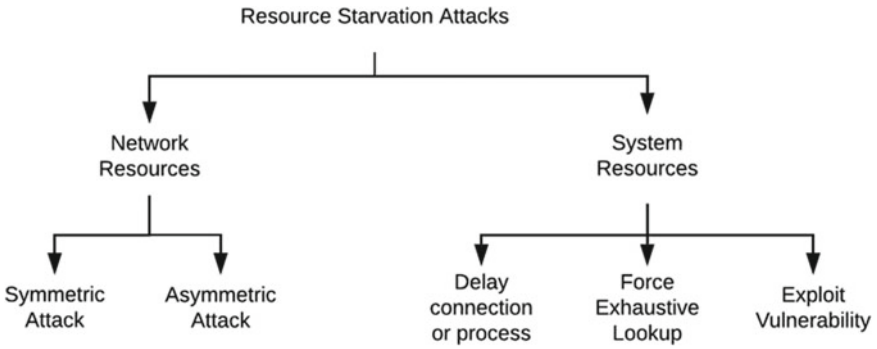

Fig. 9.5 Resource starvation (D)DoS attacks based on their main target

time or network bandwidth. Resource starvation attacks are easy to perform but hard to mitigate. Attackers make attack detection difficult by spoofing attack packets and utilizing multiple compromised nodes to generate the attack traffic. Additionally, using public network services, such as DNS and NTP, as a proxy to reflect and amplify attack traffic is a known approach utilized by attackers.

In this section, we focus on commonly used DDoS attacks targeting system and network resources to disable and/or disrupt distributed systems on the internet. We classify and present these attacks based on the resources they mainly target (See Fig. 9.5).

9.4.1 Network Resources

DDoS attacks targeting network resources are easy to perform and comprise a powerful way of disabling an online system. Attackers generally use network protocols, such as UDP, ICMP, HTTP and DNS, that do not authenticate the sender's identity. They flood the victim network with dummy traffic using zombie agents. The amount of attack traffic generated is proportional to the number of zombies used by the attacker. If zombies send attack traffic directly to the victim, it is called a symmetric DDoS attack. Attackers need a large number of zombie agents to perform an effective symmetric DDoS attack. In an asymmetric DDoS attack case, the attacker reflects and amplifies the attack traffic from an unprotected and misconfigured public network server; such as DNS, NTP and Memcached. Asymmetric DDoS attacks conceal the attackers' identity and amplify the attack strength based on the protocol exploited in the reflecting server (Ozcelik [11]).

9.4.2 System Resources

Hackers also attack limited resources of online systems and services to disable them. The purpose of these attacks is to consume all the CPU time, memory and HDD space of these systems and force them to deny legitimate users. Attackers use three general approaches to accomplish this goal: stalling communication or process to create long queues, forcing system for exhaustive lookups and exploiting vulnerabilities.

The TCP protocol defines how to establish a reliable connection and data exchange between two nodes. Attackers abuse certain protocol steps to stall the communication process during both the connection-establishment and data-exchange phases. In SYN Flood, the attacker sends many SYN packets to start new TCP connections at the victim server, but never follows up to complete the process. Eventually, the attacker uses all the memory space designated for TCP session records and forces the victim to deny service for legitimate users. Similarly, in low and slow communication attacks, the attackers prolong sessions as long as possible. In this case, the attacker establishes the TCP connection, but it slows down the data transfer rate to the minimum level required to keep the connection alive. The victim eventually reaches the maximum possible connection limits and the system denies the rest of the incoming connection requests. Slowloris, RUDY, and Slow read attacks are some of the examples of low and slow attacks.

A system needs to keep track of active TCP connections and perform a lookup to find the destination process when it receives a new packet. This lookup creates a bottleneck during peak hours. Attackers exploit this inherent weakness of TCP protocol in SYN-ACK Flood, ACK & PUSH ACK Flood and Fragmented ACK attacks. In SYN-ACK and ACK & PUSH ACK attacks, the victim server is bombarded with dummy SYN-ACK and ACK packets and overwhelmed with non-existing session lookups. In Fragmented ACK attack, the attacker sends ACK packets larger than network MTU. Therefore, the victim server deals with the defragmentation process in addition to TCP session lookups. Attackers use spoofed TCP FIN and TCP RST packets for the same purpose.

Attackers also exploit vulnerabilities of systems and protocols to perform denial of service. Fragmentation & Reassembly, killapache, and Local Area Network Denial (LAND) attacks are some of the examples in this category. In these attacks, attackers exploit a vulnerability to consume all available resources of the system. Ping of Death is an outdated attack that leveraged a vulnerability in the network stack. The attacker sent packet fragments that were larger than the system could handle after reassembly. Teardrop is also one of the more famous attacks, which targets the TCP/IP reassembly mechanism. In this attack, the attacker specially crafts packet fragments whose offset values overlap. This overwhelms the target during reassembly and causes it to fail. Similarly, a perl script released by a security researcher, whose screen name is Kingcope, sends specially crafted HTTP GET requests to exhaust the CPU and system memory of vulnerable Apache servers (Gulik [16]). In a LAND attack, the attacker specially crafts an SYN packet with the same source and destination

IP address. When the victim responds to the request, it creates an infinite loop that eventually causes the victim to crash.

9.4.3 DDoS Mitigation

For effective DDoS mitigation, attack detection and reaction systems need to work together. While efficient DDoS attack detection can be done at the attack target, reaction systems should be placed closer to the attack source. This requires a distributed system design in DDoS mitigation.

Today, most of the efficient and practical DDoS mitigation systems utilize contemporary networking and cloud technologies. Instead of reacting to DDoS attacks on premise by packet filtering using Firewalls and IDSs, attack reaction systems are moved to the cloud. Many companies, such as Cloudflare, Akamai and Arbor Networks, optimized their cloud infrastructure for DDoS reaction. These infrastructures, also called scrubbing centres, are used to separate attack traffic from the legitimate traffic. These companies claim that they can reroute victim traffic to their scrubbing centre and react to DDoS attacks in almost real time. The cost of this service depends on the size of victim service or network. In 2011, Verisign charged an average of $500,000 annually to large corporations for their DDoS mitigation service (Osborne [17]). In 2017, single attack mitigation was expected to cost around $2.5 Million (Osborne [17]). There are also DDoS mitigation solutions available for small and medium-sized businesses. By using on demand elastic cloud systems, many systems were developed, such as Deflect (Deflect [18]) and DDM (Mansfield [19]), to increase availability and reduce response time of a system. These systems aim to dissipate the overwhelming impact of DDoS attack by increasing the attack surface.

Using game theory, researchers have also proposed moving target-based DDoS mitigation approaches (Brooks [20], Dingankar [21], Venkatesan [22], Wright [23]). Moving Target Defense (MTD) approaches continuously change system configuration to reduce or else to move the attack surface. These configuration changes are described as a two-player game between an attacker and a defender and the most viable change is chosen by defender to make a successful attack difficult (Ozcelik [11]). Although, these approaches are mostly theoretical, they pave the way to build and defend performable distributed systems on the internet.

9.5 Summary

This chapter's goal is to provide practical guidelines for creating performable systems. We provide the following insights:

- Concepts from game theory should be integrated into system design.
- System testing should include intentional disruptions to verify performability.
- The internet can provide many tools that can be leveraged for testing a given system's ability to adapt to disruption.

Using these concepts, it is possible to create systems that adapt well under most condition.

References

1. Myerson, R. B. (1997). *Game theory: Analysis of conflict*. Harvard University Press.
2. Shubik, M. (1982). *Game theory in the social sciences* (Vol. 1). The MIT Press.
3. Leven, S. (1996). Models of market behavior: Bringing realistic games to market. *In IEEE/IAFE 1996 Conference on Computational Intelligence for Financial Engineering (CIFEr)*, (pp. 41–48).
4. Chen, M., Tian, S., & Chen, P. (2015). Evolutionary game analysis between service of public library and the investment of government. In *2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics* (Vol. 1, pp. 191–194).
5. Trestian, R., Ormond, O.,& Muntean, G. (2012). Game theory-based network selection: Solutions and challenges. *IEEE Communications Surveys Tutorials, 14*(4), 1212–1231, Fourth 2012.
6. Changwon Kim, R. L. (2014). Game theory based autonomous vehicles operation. *International Journal of Vehicle Design, 65*(4), 360–383.
7. Li, C. (2013). Z-test of computer-and-internet-aided multimodal English listening and speaking. In *2013 IEEE Third International Conference on Information Science and Technology (ICIST)* (pp. 98–101).
8. Bo, P., Hai, L., & Xing, J. (2009). A new sampling test method for maximum maintenance time of normal distribution items. In *2009 IEEE 10th International Conference on Computer-Aided Industrial Design Conceptual Design* (pp. 2210–2212).
9. Hegde, V., Pallavi, M. S. (2015). Descriptive analytical approach to analyze the student performance by comparative study using z score factor through r language. In *2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)* (pp. 1–4).
10. Nash, J. F. Equillibrium points in N-person Games. *Proceedings of the National Academy of Sciences of the United States of America, 36*(1), 48–49.
11. Ozcelik, I., & Brooks, R. (2020). *Distributed denial of service attacks: Real-world detection and mitigation.*
12. Kan, L., Wei, Y., Hafiz Muhammad, A., Siyuan, W., Linchao, G., Kai, H. (2018). A multiple blockchains architecture on inter-blockchain communication. In *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)* (pp. 139–145).
13. Miller, D. (2018). Blockchain and the internet of Things in the industrial sector. *IT Professional, 20*(3), 15–18.
14. Fiaidhi, J., Mohammed, S., & Mohammed, S. (2018). EDI with blockchain as an enabler for extreme automation. *IT Professional, 20*(4), 66–72.
15. Samaniego, M., & Deters, R. (2016). Blockchain as a service for IoT. In *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)* (pp. 433–436)
16. van Gulik, D. -W. (2011). Retrieved from https://mail-archives.apache.org/mod_mbox/httpdannounce/201108.mbox/<20110824161640.122D387DD@minotaur.apache.org>.

17. Osborne, C. (2017). *The average DDoS attack cost for businesses rises to over $2.5 million*, ZDNet. Retrieved from https://www.zdnet.com/article/the-average-ddos-attack-cost-for-businesses-rises-to-over-2-5m/.
18. eQualit.ie. *Deflect,* Deflect. Retrieved from https://deflect.ca/#about.
19. Mansfield-Devine, S. (2011). DDoS: Threats and mitigation. *Network Security, 2011*(12), 5–12.
20. Brooks, R. R. (2004). *Disruptive security technologies with mobile code and peer-to-peer networks.* CRC Press.
21. Dingankar, C., & Brooks, R. R. (2007). Denial of service games. In *Proceedings of the Third Annual Cyber Security and Information Infrastructure Research Workshop* (pp. 7–17).
22. Venkatesan, S., Albanese, M., Amin, K., Jajodia, S., Wright, M. (2016). A moving target defense approach to mitigate DDoS attacks against proxy-based architectures. In *2016 IEEE Conference On Communications And Network Security (CNS)*, (pp. 198–206). IEEE.
23. Wright, M., Venkatesan, S, Albanese, M., & Wellman, M. P. (2016). Moving target defense against DDoS attacks: An empirical game-theoretic analysis. In *Proceedings of the 2016 ACM Workshop on Moving Target Defense* (pp. 93–104).

Naazira B. Bhat is a second-year Master's student of Holcombe Department of Electrical & Computer Engineering, Clemson University, USA and is working as a Research Assistant for Dr. Richard Brooks. She is currently working on an NSF-funded project 'Provenance Assurance using Cryptocurrency Primitives' and her work focuses on evaluating the reliability and availability of the proposed blockchain technology when exposed to cyberattacks like the D/DoS. Naazira is currently doing her second Master's degree. She received her first Master's degree from SMVD University, Jammu, India and Bachelor's degree from the University of Kashmir, India with major in Electronics and Communication Engineering. Her research interests include cybersecurity, network traffic analysis, system security and blockchain.

Dulip Madurasinghe is a second-year Ph.D. student of Department of Electrical & Computer Engineering, Clemson University. Research Assistant for Dr. Brooks working on control resiliency analysis based on game theory under the objective of distributed cyber-physical system robustness. His research interests are cybersecurity for smart grid and autonomous vehicle. He was an embedded engineer for Atlas Labs (pty) Ltd. from 2017 to 2018. He received his Bachelor from University of Moratuwa, Sri Lanka major in Electrical Engineering in 2017.

Ilker Ozcelik received the MS degree in electrical engineering from Syracuse University in 2010, and the Ph.D. degree in electrical engineering from the Holcombe Department of Electrical and Computer Engineering, Clemson University, Clemson, South Carolina in 2015. His research interests include network traffic analysis, network security, software-defined networking, blockchain, security and privacy in intelligent systems.

Richard R. Brooks has in the past been PI on research programs funded by the Air Force Office of Scientific Research, National Science Foundation, Department of Energy, National Institute of Standards, Army Research Office, Office of Naval Research and BMW Corporation. These research projects include coordination of combat missions among autonomous combat vehicles (ARO), situation and threat assessment for combat command and control (ONR), detection of protocol tunnelling through encrypted channels (AFOSR), security of intelligent building technologies (NIST), experimental analysis of Denial of Service vulnerabilities (NSF), mobile code security (ONR) and security analysis of cellular networks used for vehicle remote diagnostics (BMW). He received his BA in mathematical sciences from Johns Hopkins University and Ph.D. in computer science from Louisiana State University.

Ganesh Kumar Venayagamoorthy is the Duke Energy Distinguished Professor of Power Engineering and Professor of Electrical and Computer Engineering and Automotive Engineering at Clemson University. Prior to that, he was a Professor of Electrical and Computer Engineering at the Missouri University of Science and Technology (Missouri S&T), Rolla, USA from 2002 to 2011 and Senior Lecturer in the Department of Electronic Engineering, Durban University of Technology, Durban, South Africa from 1996 to 2002. Dr. Venayagamoorthy is the Founder (2004) and Director of the Real-Time Power and Intelligent Systems Laboratory (https://rtpis.org). He holds an Honorary Professor position in the School of Engineering at the University of Kwazulu-Natal, Durban, South Africa. Dr. Venayagamoorthy received his Ph.D. and M.Sc. (Eng) degrees in Electrical Engineering from the University of Natal, Durban, South Africa, in February 2002 and April 1999, respectively. He received his BEng (Honors) degree with a First Class from Abubakar Tafawa Balewa University, Bauchi, Nigeria in March 1994. He holds an MBA degree in Entrepreneurship and Innovation from Clemson University, SC (2016). Dr. Venayagamoorthy's interests are in the research, development and innovation of smart grid technologies and operations, including computational intelligence, intelligent sensing and monitoring, intelligent systems, integration of renewable energy sources, power system optimization, stability and control, and signal processing. He is an inventor of technologies for scalable computational intelligence for complex systems and dynamic stochastic optimal power flow. He has published over 500 refereed technical articles. His publications are cited ~17,000 times with an h-index of 63 and i10-index of 255. Dr .Venayagamoorthy has been involved in over 75 sponsored projects in excess of US $12 million. Dr. Venayagamoorthy has given over 500 invited keynotes, plenaries, presentations, tutorials and lectures in over 40 countries to date. He has several international educational and research collaborations. Dr. Venayagamoorthy is involved in the leadership and organization of many conferences including the General Chair of the Annual Power System Conference (Clemson, SC, USA) since 2013, and Pioneer and Chair/co-Chair of the IEEE Symposium of Computational Intelligence Applications in Smart Grid (CIASG) since 2011. He is currently the Chair of the IEEE PES Working Group on Intelligent Control Systems, and the Founder and Chair of IEEE Computational Intelligence Society (CIS) Task Force on Smart Grid. Dr Venayagamoorthy has served as Editor/Guest Editor of several IEEE Transactions and Elsevier Journals. Dr Venayagamoorthy is a Senior Member of the IEEE and a Fellow of the IET, UK and the SAIEE.

Anthony Skjellum studied at Caltech (BS, MS, Ph.D.). His Ph.D. work emphasized portable, parallel software for large-scale dynamic simulation, with a specific emphasis on message-passing systems, parallel non-linear and linear solvers, and massive parallelism. From 1990 to 1993, he was a computer scientist at the Lawrence Livermore National Laboratory focusing on performance-portable message passing and portable parallel math libraries. From 1993 to 2003, he was on the faculty in Computer Science at Mississippi State University, where his group coinvented the MPICH implementation of the Message Passing Interface (MPI) together with colleagues at Argonne National Laboratory. From 2003–2013, he was professor and chair at the University of Alabama at Birmingham, Department of Computer and Information Sciences. In 2014, he joined Auburn University as Lead Cyber Scientist and led R&D in cyber and High-Performance Computing for over three years. In Summer 2017, he joined the University of Tennessee at Chattanooga as Professor of Computer Science, Chair of Excellence, and Director, SimCenter, where he continues work in HPC (emphasizing MPI, scalable libraries, and heterogeneous computing) and Cybersecurity (with strong emphases on IoT and blockchain technologies). He is a senior member of ACM, IEEE, ASEE, and AIChE, and an Associate Member of the American Academy of Forensic Science (AAFS), Digital & Multimedia Sciences Division.