Article

N-Versions-Based Resilient Traffic Control Systems

Abdullah Basuhail *, Maher Khemakhem, Fathy Elbouraey Eassa, Junaid Mohammad Qurashi * and Kamal Jambi

Department of Computer Science, Faculty of Computing and Information Technology, King Abdulaziz University (KAU), Jeddah 21589, Saudi Arabia; makhemakhem@kau.edu.sa (M.K.); feassa@kau.edu.sa (F.E.E.); kjambi@kau.edu.sa (K.J.)
* Correspondence: abasuhail@kau.edu.sa (A.B.); jqurashi0001@stu.kau.edu.sa (J.M.Q.)

Abstract: Increasing the resilience of traffic control systems is a priority for many important cities worldwide. This is due to the ever-increasing problems leading to different failures in such systems. We are witnessing the intensive introduction of new technologies that automatically manage traffic but are exposed to different kinds of attacks. There are also unpredictable increases in climatic changes and the number of cars in many cities. These factors will surely enhance the failure risks of such systems and consequently increase the damage caused by traffic jams and road accidents. In this paper, we introduce a resilient traffic control system that consists of three levels: sensor control, display, and light control. Each level has three (or more) versions and a dynamic voter. Hence, the introduced system is based on diversity and redundancy (replication), called N-versions. We propose two techniques for the introduced resilient traffic control system. The first technique uses N-versions and dynamic voters to vote between the outcomes in each level. The second technique uses N-versions, dynamic voters, and acceptance testing units. The overhead in the second technique is evidently greater than that of the first technique, but its resilience is better. A fine analytical study is conducted and shows that the first technique requires only three versions to reach the optimal results, bounded by 1/15 probability of having a faulty system. The second technique leads to better results, which can determine small probabilities.

Keywords: failure analysis; N-versions software; resilience; risk management; software reliability; traffic control

check for updates

Citation: Basuhail, A.; Khemakhem, M.; Eassa, F.E.; Qurashi, J.M.; Jambi, K. N-Versions-Based Resilient Traffic Control Systems. *Electronics* **2022**, *11*, 2414. https://doi.org/10.3390/electronics11152414

Academic Editors: Javier Alonso Ruiz and Angel Llamazares

Received: 21 June 2022
Accepted: 28 July 2022
Published: 2 August 2022

1. Introduction

A traffic control system is a part of any road infrastructure. Having a safe road infrastructure in any given city and/or place is the ultimate objective of any government and the expectation of any road user. The safety of the road infrastructure is an important indicator of the prosperity of nations. Thus, engineers and researchers specializing in traffic problems have long been working to improve our daily life by proposing solutions and ideas to mitigate such problems and fulfill this objective and expectation. Consequently, the literature is saturated with proposed solutions, as explained in the subsequent section. Unfortunately, despite the diversity of these proposed ideas, the problem remains. Therefore, in this work, we focus on the improvement of traffic control systems that make the road infrastructure more resilient. The existing infrastructure is, unfortunately, exposed to several kinds of problems that can lead to failure and, thus, serious traffic problems. The problems that lead to these failures come from multiple sources: the intensive use of new technologies (which are not always safe and accurate) in traffic management, increases in unexpected climate changes, and terrorist attacks are the main challenges. According to some specialists [1], unexpected climate change phenomena can occur at any time and any place, leading to several kinds of damage, affecting the road infrastructure. In addition, some terrorist attacks focus on the road infrastructure.

We are now witnessing the era of smart vehicles, which can self-drive. The problem of road infrastructure failure will be further complicated if the resilience of such an infrastructure is not enhanced. Therefore, the ultimate objective of this work is to strengthen the resilience of the road infrastructure to substantially mitigate its failure in case of problems. To deal with this problem, we propose a detailed architecture, mainly based on some well-known techniques, such as replication, and a multiversion software. The proposed architecture is first discussed, evaluated, and compared with current solutions. Then, we demonstrate the added values of the architecture related to its relatively high cost. The rest of this paper is organized as follows. Section 2 provides a background of the ITS infrastructure and highlights the vulnerabilities faced in the design. Section 3 gives an overview of the state of the art related to the studied topic. Section 4 gives a brief overview of the threat model, describing the attack surfaces of the ITS. Section 5 details the proposed architecture, which is then discussed and evaluated in Section 6. The conclusion and future work are given in Section 7.

2. Background

2.1. Intelligent Transportation System

Cyber–physical systems (CPSs), including intelligent transportation systems (ITSs), will be the lifeline of smart cities in future. Traffic management is going to be an important part of the ITS. The underlying technology—i.e., vehicular ad-hoc networks (VANETs)—provides a means for vehicles to intelligently exchange messages regarding road and traffic conditions to enhance safety. The open nature of ITS as a wireless communication technology leads to threats and vulnerabilities to cyber-attacks. A proposed taxonomy of security and privacy issues and solutions in ITS was addressed in [2]. A detailed survey of attacks and a risk analysis of the vulnerabilities against TCS was completed in the study [3]. A false data injection attack on TCS was carried out in [4], while Gurcan Comert et al. [5] assessed cyber-attacks against intelligent traffic signal systems and devised a vulnerability score.

In ITS, with the growing emphasis on self-driving vehicles (SDC), the infrastructure needed to support such a technology will be heavily dependent on the roadside infrastructure, including traffic controllers and roadside units (RSUs). An overview of the ITS infrastructure is given in Figure 1. Further, the essential connectivity of the SDC with the road infrastructure and other vehicles on the road makes ITS highly vulnerable to cyber-attacks, thus exposing many attack surfaces. Vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), and in-vehicle communication will provide adversaries with plentiful opportunities to exploit the vulnerabilities in the design and architecture of the entire ITS. Exploitation can be achieved both remotely and locally. For example, in V2V communication, vehicles communicate with each other through a shared network. Any malicious vehicle in a V2V network will impose a threat on the entire infrastructure, and its aftermath would have a cascading effect to all the shared and connected networks. Similarly, any compromise in the cellular networks that connects to the ITS infrastructure, such as traffic control towers (which house traffic controllers and traffic management systems to regulate traffic through connected RSUs) can have unpredictable outcomes. For example, adversaries were able to take control of the Jeep Cherokee through vulnerabilities found in the cellular networks [6]. Various incidents of attacks against ITS have been reported, including specific attacks on unmanned vehicles [7]. Further, several proof-of-concept works were explored that expose the vulnerabilities in ITS.

Evidently, achieving safety and reliability in such systems is still far fetched due to the cyber threats posed by the vulnerabilities in the ITS infrastructure. Although several works have tried to address the issue using traditional measures of risk assessment and cybersecurity [8], the results are inefficient due to these systems' dependence on continuous cyber-connectivity and the rate at which threats against such systems are devised. This makes it nearly impossible to completely safeguard these systems. Thus, due to the high probability of cyber-attacks occurring in existing or emerging cyber–physical systems, these systems should be redesigned to have resilience.

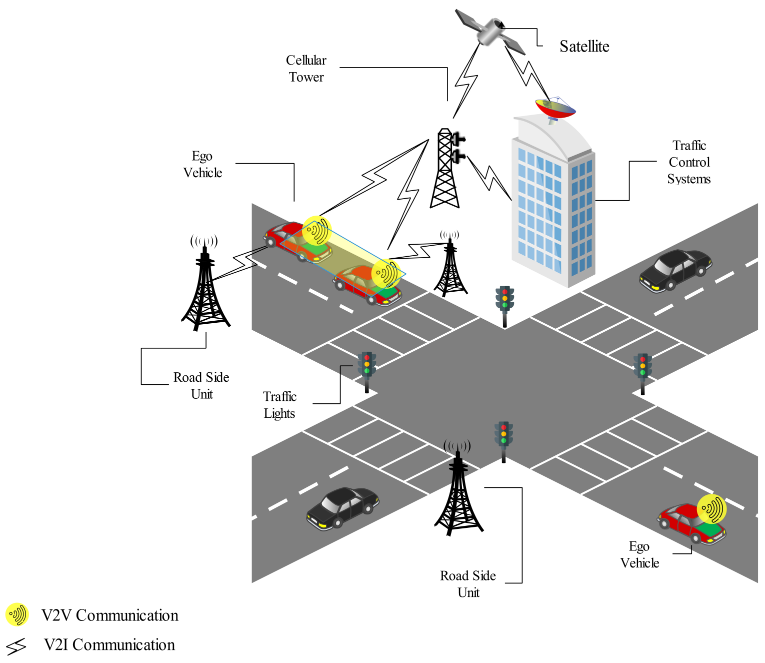

Figure 1. Overview of the ITS infrastructure.

Resilience

Resilience, although a new concept to the field of engineering, has been one of the primitive features of natural sciences such as ecology, economics, social sciences and management sciences. Engineering and computer science are other recent fields that have incorporated the concept of resilience [9]. Resilience, in this context, is defined as the system's ability to function at the desired level after having suffered a partial damage. Some of the foremost traits of resilience are redundancy and diversity. These can be considered core traits of resilience, although other traits, such as self-healing, adaptation, etc., also have been considered to be conceptually feasible [10]. However, in this research, we adhere to the concept of resilience as having diversified and redundant units measuring the same output. The assumption is that having diversified and redundant units will make it difficult for the adversary to compromise all the units at the same time. Further, the assumption in a conservative resilience model is that, at any given time, given that the number of redundant units is n, the number of compromised units will be less than $(n-1)/3$. Having redundant units will ensure a good performance despite some of the redundant units being compromised. Resilient solutions are independent of the type of attack, unlike traditional cyber-security measures, which are devised by considering a particular attack.

It is difficult to prevent all attacks, but redesigning systems for resilience will make attacks less likely to damage systems [1]. Moreover, in cases of successful attacks, resiliency will minimize the consequences and increase the costs and uncertainties in attacking, possibly preventing fall-out [1]. Resilience can be considered an attribute of dependability [1], which has been used for decades as a means of providing fault tolerance in the information technology and communication world [11]. However, addressing these issues using the traditional measures, as previously mentioned, will not suffice. However, although few works have included resilience in their works, there has not been much focus on TCS. The following section section provides a review of the resilient techniques employed in ITS in general and, if possible, on TCS.

3. Related Work

The authors of [8] modeled an attack on TCS to create different road traffic scenarios and evaluated the severity of disruptions caused at different intersections in urban areas. The contribution of the work lies in their finding a correlation between the efficiency and resilience, which was found to be non-significant. However, the work's explanation of how the resilience was modeled into their TCS architecture is lacking; hence, the evaluation is based on the assumption that TCS is resilient.

A similar approach to finding a correlation between vulnerability and resilience in an air traffic control (ATC) system was used [4]. The assessment was mostly based on network characteristics related to topology. The authors evaluated the resilience of the ATC against random and deliberate attacks. Deliberate attacks included *degree attacks* and *betweenness attack*.

The recovery techniques that were employed included discarding the affected paths and nodes while connecting to different paths and nodes until the affected paths and nodes were restored to their normal functioning. The prioritization of nodes to be recovered determined the strategy employed to recover the network. The limitations were attributed to their only considering node failure; the technique does not consider attacks on data or information flow, which, in the scenario, are most likely to occur. Overall, resilience had a negative correlation with the betweenness attack and had a significant impact on the overall functionality of the network. A similar approach was utilized by Kevin L. Clark et al. [12] to determine the characteristics and configurations for a resilient air traffic flow against cyber-attacks or any other disruption. Some researchers integrated evolutionary algorithms and a novelty search to improve the performance and resilience of autonomous systems and have developed tools for this purpose [13]. The use of infrastructure-to-vehicle (I2V) communication to generate routing suggestions for drivers in transportation systems, with the goal of optimizing a measure of overall network congestion, was addressed in [14].

There have been considerably fewer works on resilient traffic control systems; however, generally, resilient techniques have been developed for controllers in other domains. For example, in [15], to safeguard microgrids against stealthy attacks, the authors proposed a redundant controller policy, which would be switched upon false data injection attacks. The technique was developed using the Luenberger observer and unknown input compensator, which optimize the disturbed signal, making it a stable control signal.

Based on a very similar technique, the authors in [16] proposed a redundant resilient controller strategy for wireless–sensor–actuator networks that switch the controller upon failure. Further, the authors devised a protocol that would limit the delay upon switching the controllers. However, the study limitations lie in the assumption that the second controller is devoid of any failures, and cyber-attacks that could cause deliberate disruption have not been considered. The authors in [17] proposed a resilient optimization method to avoid traffic congestion owing to different disruptions. However, the work only focused on optimizing the signals and disregarded any probability of cyber-attacks.

4. Threat Model

The intelligent transportation system, just like any other cyber–physical system, will be exposed to external and internal vulnerabilities that could be exploited to gain accessibility, lose availability or confidentiality, and compromise the safety and reliability of ITS systems. Satellite communication, cellular towers, control towers, RSUs, over-the-air-updates, sensors, vehicle infotainment systems, including Wi-Fi and Bluetooth, diagnostic ports, such as OBD-II ports of cars, including SDC, and vehicle communication channels such as controller area network (CAN) or FlexNet all provide attack surfaces that adversaries can exploit for malicious reasons. Attackers' actions could lead to disruptions in the normal functioning of ITS and its connected entities and, in more serious cases, cause fatal incidents that lead to the loss of human life.

The attacker could be anyone who exploits any of the below-mentioned attack surfaces, with motives ranging from self-gratification to financial gain. The attacker profile could be that of an individual or group of individuals working in an organized manner to cause serious damage to ITS assets. The sets of attacks that could be devised are code injection/modification, spoofing attacks, replay attacks, relay attacks, side channel attacks, jamming Attacks, DoS, DDoS, and adversarial attacks against any element of ITS, as described above.

- In-vehicle communication: One of the primary targets of the attackers would be to manipulate a vehicle on the road. Any direct access to the vehicle through ports, such as OBD-Ports, would allow for considerable access to the core functionalities of the vehicle. Multiple controllers could be compromised through code-injection or code-modification attacks. Any compromise in the in-vehicle communication would lead to attacks and have a cascading effect on the entire ITS infrastructure. Data from sensors and electronic controller units (ECUs) could be intercepted and manipulated for further attacks. Although local access to OBD-II Ports is harder in real-world settings, it is highly likely that adversaries could achieve remote access to in-vehicle communication.
- Vehicle-to-Vehicle: V2V communication of the vehicles through a shared network should allow for a smoother and better experience on the road. However, the shared network can be manipulated for transitory attacks that would have a cascading effect on other vehicles on the road. Any malicious information shared among the vehicles, such as road conditions or obstacles, if wrong or fake, could have serious consequences.
- Vehicle-to-Infrastructure: Finally, the core back-end communication network relies on the cellular network and satellite communication. If exploited, this could offer widespread accessibility to various controlling entities in the infrastructure. Any compromise in the network would mean more serious consequences and a widespread aftermath. Compromised RSUs, cellular networks and traffic control systems could pave the way for adversaries to cause more concrete damage.

However, irrespective of the attack being devised, resilient solutions are meant to ensure partial or full recovery of the functionality of the ITS within a stipulated amount of time. As resilient solutions already assume that the system or plant has been attacked, irrespective of the nature of the attack, a resilient-based system should be able recover functionality. The idea behind the resilient solutions is to grant optimal functionality to the system, despite any faults or attacks, as, with the ever-growing rate of cyber-attacks, ITS is going to be an easy target in the near future, and with the pace and sophistication with which these attacks are devised, it is nearly impossible to provide a counter-solution to each and every threat being constructed, hence the adoption of resilient solutions.

5. Proposed System

In this paper, we introduce some software techniques to increase the resilience of the traffic control system, which is part of the road infrastructure. A multiprocessor traffic control system, as shown in Figure 2, consists of three processes: a sensor control process, display process, and light control process. The three processes may be executed on different processors. The system can be attacked by an external attack or internal attack, and the system will fail if one process is infected (corrupted) by the attack. The resilience and availability of the system are low.

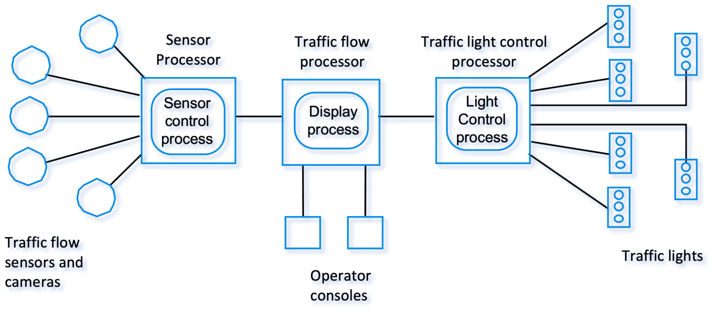

Figure 2. A multiprocessor traffic control system.

6. Resilient Traffic Control System

6.1. Resilient Traffic Control System Based on Dynamic Voters

Resilient Traffic Control System Architecture

We introduce a technique to increase the resilience of the system. The technique is based on diversity and redundancy concepts. Figure 3 illustrates the architecture of the resilience technique. In the proposed resilience technique, there are three versions for each process, as shown in Figure 3. The three versions of the display process (master process) first run by sending polling signals to versions of the sensor control process (processors), then the three polling signals are gathered and accepted by the voter. The voter produces a correct polling signal or an incorrect signal. The correct polling signal is sent to each version of the sensor control process (slave-process) to begin execution. The three versions of the sensor control process receive correct polling signals from the voter to simultaneously run on three processors (or cores). The three versions of the sensor control process periodically poll the sensors to capture information on traffic flow and collate this information for further processing. The outcomes of the three versions of the sensor control process are gathered by the resilience technique and sent to dynamic voter#1. The dynamic voter votes between two or three outcomes (i.e., the fastest outcomes should be compared by the dynamic voter). Based on the Algorithm 1 the decision is taken by the voter#1:

Algorithm 1 Comparing sensor control process output.

```
Require: Output from three different sensors
Ensure: Comparisons of outputs
1:  if (outcome1 = outcome2) then
2:      correct-outcome = outcome1 // sent to Display process//
3:  end if
4:  if (outcome1 != outcome2) then
5:      the voter compares between outcome1 and outcome3 or outcome2 and outcome3
6:  end if
7:  if (outcome1 = outcome3 ) then
8:      correct-outcome = outcome3 //sent to Display process//
9:  end if
10: if (outcome2 = outcome3 ) then
11:     correct-outcome = outcome3 //sent to Display process//
12: end if
13: if (outcome1!= outcome2 != outcome3) then
14:     Fail!
15: end if
```

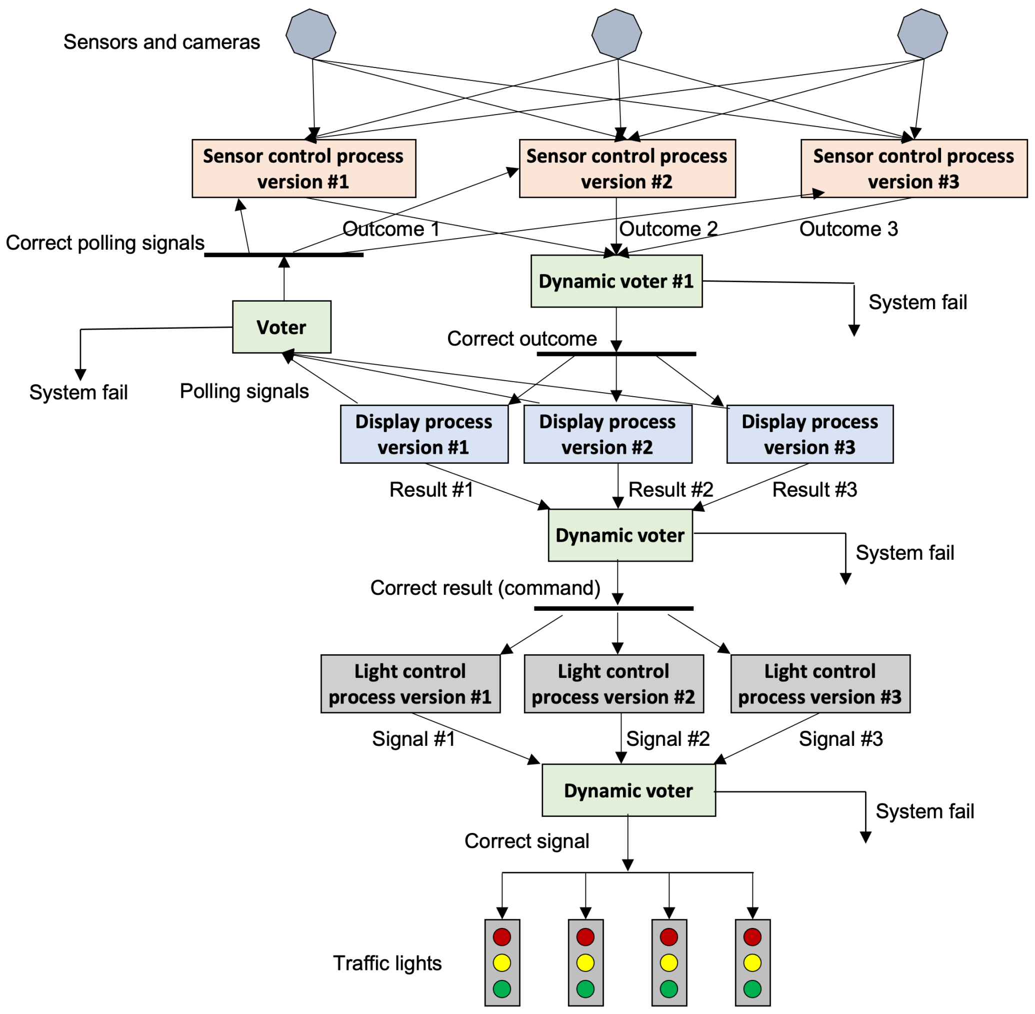

Figure 3. Resilient multiprocessor traffic control system.

The correct outcome is sent to three versions of the display process (master process) to simultaneously run on three parallel processors or cores. The three versions of the master process are concerned with displaying traffic status to the operators, computing the traffic light sequences, and accepting operator commands to modify the light sequences. Commands (results) sent by the three versions of the master process to traffic light control process are gathered by resilience technique and voted on by voter#2. The fastest two commands are gathered by the resilience technique and sent to dynamic voter#2. The voter takes the decision based on the following Algorithm 2:

Algorithm 2 Comparing display process output.

```
Require: Output from three different display processes
Ensure: Comparisons of outputs
 1: if (result1 = result ) then
 2:     correct-result = result1 // sent to light control process
 3: end if
 4: if result1 != result2  then
 5:     the voter compares between result1 & result3 OR result2 && result3
 6: end if
 7: if (result1 = result3) then
 8:     correct-result = result3 // Light control process
 9: end if
10: if (result2 = result3) then
11:     correct-result = result3 // Light control process
12: end if
13: if (result1 != result != result3) then
14:     system fail
15: end if
```

The correct result (command) is sent to the three versions of the light control process to run in parallel on three cores or processors. Each version converts the received command into signals to control the traffic light hardware. The fastest two signals are collected and sent to the dynamic voter#3. The voter takes the decision according to following Algorithm 3:

Algorithm 3 Comparing light controller process output.

```
Require: Output from three different light control processes
Ensure: Comparisons of outputs
 1: if (signal1 = signal2 ) then
 2:     correct - signal = signal1 // sent to traffic light
 3: end if
 4: if signal1!= signal2  then
 5:     the voter compares between signal1 & signal3 OR signal2 && signal3
 6: end if
 7: if (signal1 = signal3) then
 8:     correct - signal = signal3 // sent to traffic light
 9: end if
10: if (signal2 = signal3) then
11:     correct-signal = signal3 // sent to traffic light
12: end if
13: if (signal1 != signal2!= signal3) then
14:     system fail // sent to traffic light
15: end if
```

The correct signal is sent to light hardware, or system failure will occur if there is no correct signal.

6.2. Overhead of the Resilience Technique

The size overhead can be computed as follows:

Size overhead = $(level\#1 - size - overhead) + (level\#2 - size - overhead) + (level\#3 - size - overhead) = (sizeoftwosensor - control - process - versions + sizeofdynamicvoter\#1) + (sizeoftwodisplay - process - versions + size - of - polling - voter + size - of - dynamic - voter\#1) + (sizeoftwolight - control - process - versions + size - of - dynamic - voter\#3$

The time overhead can be computed as follows:

Time overhead = $(level\#1 - time - overhead) + (level\#2 - time - overhead) + (level\#3 - time - overhead) = ((slowest - sensor - process - version - time - fastest - sensor - process - version - time) + gathering - outcomes - time + dynamic - voter\#1 - time) + ((slowest - display - version - time - fastest - display - version - time) + gatheringtime + polling - voter - time + dynamic - voter\#2 - time) + ((slowest - light - control - version - time - fastest - light - control - version - time) + gathering - time + dynamic - voter\#3 - time)$

7. Resilient Traffic Control System Based on Voting and Acceptance Testing

As shown in Figure 4, the outcomes of the three versions of the sensor control process are gathered and voted on by voter#1 according to the rules mentioned above. If there is a correct outcome, it is sent to three versions of the display process. If an incorrect outcome is obtained from voter#1, the acceptance test (AT) module receives outcome1 to ensure correctness. If outcome1 is correct according to AT, it is sent to three versions of the display process. If outcome1 is not accepted by AT, then outcome2 is sent to AT for testing. If outcome2 is accepted by AT, it is sent to the three versions of the display process; otherwise, outcome3 is sent to AT for testing. If outcome3 is accepted by AT, it is sent as a correct output to the three versions of the display process; otherwise, the system has failed.

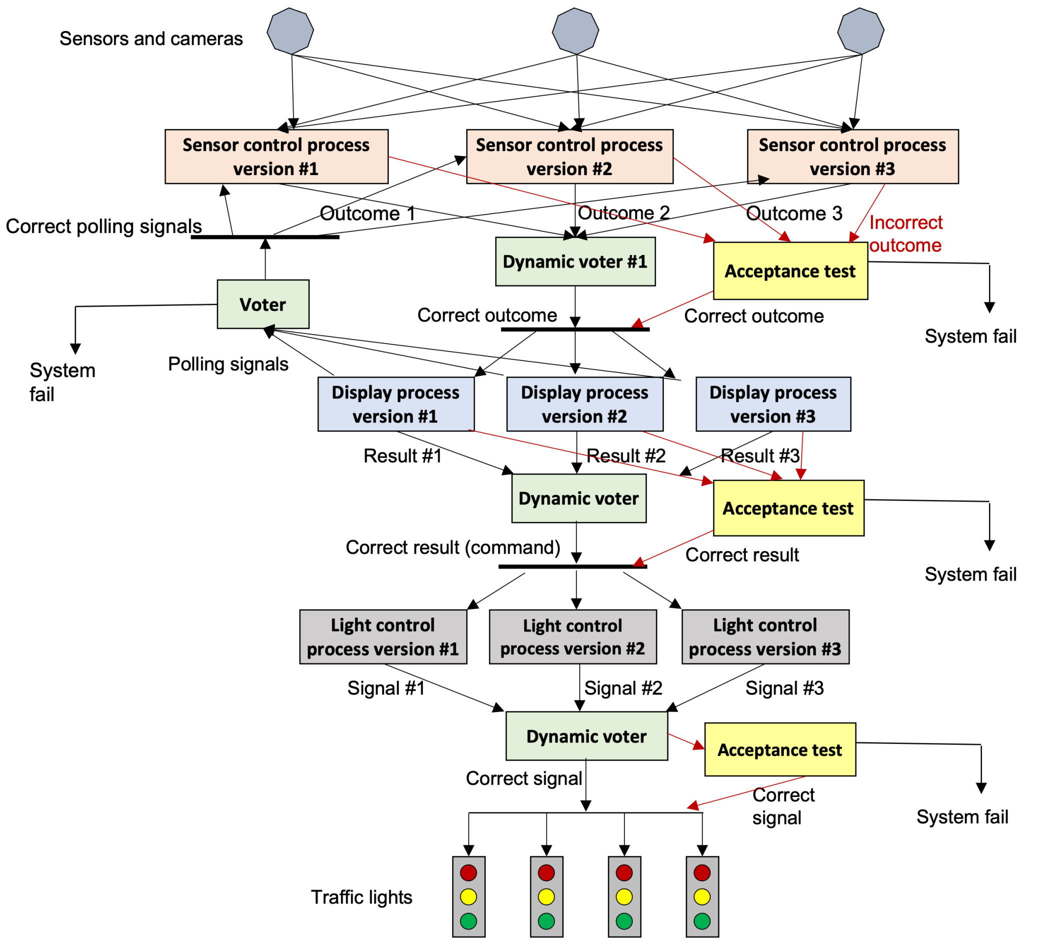

Figure 4. Resilient multiprocessor traffic control system based on voter and acceptance tests.

The display results are voted on by the dynamic voter#2. The correct result is directly sent to the three versions of the light control process. If the output of the voter#2 is an incorrect result, the three results are checked by an acceptance test (AT). Any outcome that passes the AT is considered the correct outcome and sent to the display process versions. The results of the display process versions are gathered and sent to dynamic voter#2. If the voter provides an incorrect result, the three results are individually verified by the acceptance test (AT) module. Any result that passes the AT is considered the correct result. If no result passes the AT, the system fails. The correct result is sent to the light control process versions. The signals of these versions are gathered and sent to dynamic voter#3.

Overhead of the Voting and Acceptance Testing Resilience Techniques

The overhead of the previously voting acceptance techniques can be computed as follows:

Size overhead = $(level\#1 - size - overhead) + (level\#2 - size - overhead) + (level\#3 - size - overhead) = (size - of - two - sensor - control - process - versions + size - of - dynamic - voter\#1 + acceptance - testing - size) + (size - of - two - display - process - versions + size - of - polling - voter + size - of - dynamic - voter\#2 + acceptance - testing - size) + (size - of - twolight - control - process - versions + size - of - dynamic - voter\#3 + acceptance - testing - size)$

The time overhead can be computed as follows:

Time overhead = $(level\#1 - time - overhead) + (level\#2 - time - overhead) + (level\#3 - time - overhead) = ((slowest - sensor - process - version - time - fastest - sensor - process - version - time) + gathering - outcomes - time + dynamic - voter\#1 - time + [\sum_{k=1}^{n} acceptance - testing - time]) + ((slowest - display - version - time - fastest - display - version - time) + gatheringtime + polling - voter - time + dynamic - voter\#2 - time[\sum_{k=1}^{n} acceptance - testing - time] + ((slowest - light - control - version - time - fastest - light - control - version - time) + gathering - time + dynamic - voter\#3 - time[\sum_{k=1}^{n} acceptance - testing - time])$, where $n = 1, 2$, or 3.

8. Cause of No Outcome Being Produced for Any Software Version

Any version in the three levels of the two resilient techniques can produce no outcome. The cause of no outcome may be a fault in the software version itself or a fault in the primary processor (core). To determine the cause of no outcome, the resilient technique conducts the following:

1. Move the version to another no-faulty secondary processor for execution.
2. If there is no outcome, this means the version is the faulty version.
3. If there is an outcome, this means the primary processor is faulty.
4. The cause of the error is repaired.

9. Evaluation of the Strengths and Weaknesses of the Proposed System

As depicted in Figures 2 and 3, we consider the following notations:

1. SVi denotes sensor control version i, where $1 \leq i \leq 3$, and we consider that this level corresponds to level 1, denoted by $L1$.
2. DVi denotes display processed version i, where $1 \leq i \leq 3$, and we consider that this level corresponds to level 2, denoted by $L2$.
3. LVi denotes light control version i, where $1 \leq i \leq 3$, and we consider that this level corresponds to level 3, denoted by $L3$.

Then, we can easily determine the probability of failure for each level denoted by PFL_i, where $1 \leq i \leq 3$ is
obtained as follows:

1. In $L1$, the system is faulty only in case of: $SV1 \neq SV2 \neq SV3$ ($SVi \neq SVj$, which means that the obtained results are different). This means that, for all the remaining cases, the system works properly, and this is reached when $SV1 = SV2 = SV3$

($SVi = SVj$, which means that obtained results are equal) or $SV1 = SV2 \neq SV3$ or $SV1 = SV3 \neq SV2$ or $SV2 = SV3 \neq SV1$; this confirms that PFL_1 is

$$PFL_1 = 1/5 \tag{1}$$

2. In the same manner, for the remaining levels, $L2$ and $L3$:

$$PFL_1 = PFL_2 = PFL_3 = 1/5 \tag{2}$$

Let us now compute the overall probability of failure for the entire system. We know that the proposed system only works properly if the three levels work properly (without failure); otherwise, the system is faulty. If one of these three levels is faulty, then the overall system will be faulty; this leads to the probability of a faulty level which is equal to 1/3. Hence, the probability of having a faulty system denoted by PFS_n (for only scenario 1, where n denotes the total number of used versions) will be the conditional probability of having a fault in any given level; this gives the following equation:

$$PFS_3 = 1/3 \times 1/5 = 1/15 \tag{3}$$

This means that the probability of having a fault in the proposed system is 1/15 if we consider Figure 3 and 1/60 for Figure 4, since, in this case, we added the acceptance test, which improves the resilience of the system. Consequently, the proposed system with its two scenarios (Figures 3 and 4) substantially improves the resilience of traffic control systems. We note that if we increase the number of versions (more than 3), then the probability of failure will only decrease for the second technique (with acceptance test) and vice versa (but see the corresponding cost). Indeed, the general equation, which governs this issue, is as follows:

$$PFSA_n = PFS_n \times 1/(n+1) \tag{4}$$

where:

$PFSA_n$ is the probability of having a faulty system with n versions with acceptance test;
PFS_n is the probability of having a faulty system with n versions without acceptance test;
n is the number of versions (replications) and must be an odd number $n \geq 3$.

For the acceptance test, we note that the total number of possibilities for n versions is $n+1$, which is easily obtained by the following equation:

$$TNPoA_n = \binom{n}{1} + \binom{n}{0} = n+1 \tag{5}$$

where:

$TNPoA_n$: denotes the total number of possibilities with the acceptance test when using n versions.

Only one possibility can lead to a faulty system with the acceptance test; consequently, the probability of having a faulty system with the acceptance test when using n versions is $1/(n+1)$ at each level, as mentioned in Equation (4), considering the conditional probability. We also note that the optimal number of versions for technique1 (without acceptance test) is 3; this is because if we increase this number, then the probability of failures will also be increased. As we have seen the case where the number of versions was 3, the total number of possibilities of having a faulty system is only 1, and the total number of possibilities (faulty, no faulty) is 5. These two numbers (1 for "faulty", and 5 for "faulty and no faulty") are obtained, respectively and simply, by the two following equations:

$$TNPoFS_n = \sum_{k=2}^{(n-1)/2} \binom{n}{k} + \binom{n}{0} \tag{6}$$

$$TNPoS_n = \sum_{k=2}^{n} \binom{n}{k} + \binom{n}{0} \tag{7}$$

where $TNPoFSn$ denotes the total number of possibilities of having a faulty system when using n versions, and $TNPoSn$ denotes the total number of possibilities (faulty, not faulty) when using n versions. Of course, $\binom{n}{0} = 1$.

Consequently, the probability of having a faulty system for scenario1 when using n versions is given by the following ratio:

$$PFS_n = \frac{TNPoFS_n}{TNPoS_n \times 3} \tag{8}$$

In a similar manner, the probability of having a faulty system for scenario2 when using the n versions is given by the following ratio:

$$PFSA_n = \frac{PFS_n}{(n+1)} \tag{9}$$

Figure 5 illustrates the two studied techniques, PFS_n and $PFSA_n$, where n ranges from 3 to 27.

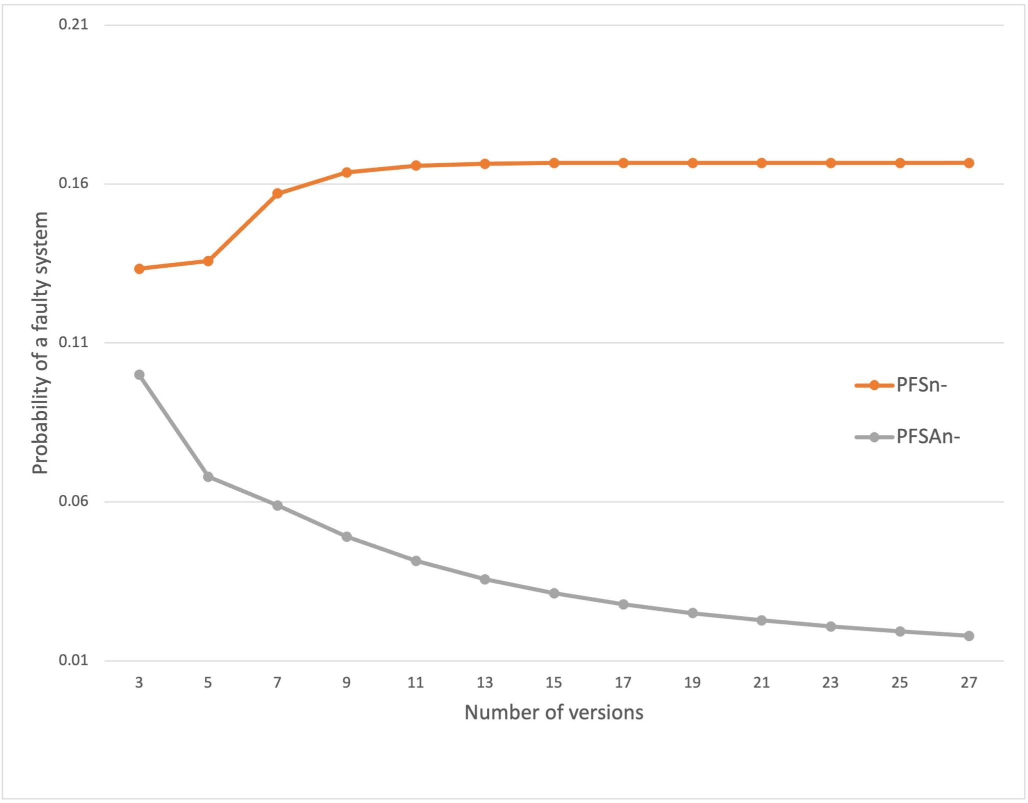

Figure 5. The two studied techniques, PFS_n and $PFSA_n$.

It is clear that, for technique1 (PFS_n), the optimal number of version n is 3, since this value provides the minimum probability of having a faulty system. We can also clearly see that when the number of versions n increases, the probability of the system being faulty also increases for technique1 (PFS_n). At n = 5 and n = 7, there is a significant increase in the probability of the system being faulty. However, with this technique, the probability of the system being faulty stabilizes over a constant value, as n is increased to 27. Interestingly, the inference from the analysis could be that replication alone does not necessarily reduce the probability of system failure, as the chances of failures in replicated units also increases. The rise in the probability of system failure from n = 5 to n = 7 exasperates this phenomenon before stabilizing to a constant value. This makes complete sense, as there is no way of determining which of the units is faulty; hence, decisions made on faulty or malicious data would reduce the functionality of the system. Increasing the number of versions to more than 9 does not necessarily further increase the probability of the system being faulty, perhaps because there are equal chances of replicated versions being faulty/malicious or not faulty/malicious; this is the reason for the flattened line on the graph after the number of versions reaches more than 9.

In contrast to this, for technique2 ($PFSA_n$), we observe that, starting from $n = 3$, the probability of having a faulty system considerably decreases as n increases, essentially decreasing to 0. This phenomenon can be attributed to having a clear indication of which replicated units are corrupt/faulty and which are not. Hence, only the correct data are forwarded from non-corrupt units. With the acceptance feature, the resilient technique, i.e., $PFSA_n$, performs better than the PFS_n technique as is also evident from the analysis. These observations lead to the following recommendations:

1. For technique1, the optimal number of versions n to implement to minimize the probability of having a faulty system is 3 if we are sure that the corresponding cost does not exceed the cost of the possible damage caused by such faults. This can be determined using an easy statistical study on the history of the different damages caused by the failure in any given traffic control system.
2. Any suitable decision on the number of versions n to implement for technique2 must be preceded by a statistical study on the history of the different damages caused by the failure of any given traffic control system.
3. The more the statistical studies on the history of the different damages caused by the failure of any given traffic control system are refined, the higher the quality of the decision will be for technique2.

Regarding the overhead of both techniques presented in the proposed system techniques, we think that the time required for the voting processes and the acceptance test in the three levels will not penalize the overall system in terms of response time because, with the new and diverse parallel processing architectures, one can easily resolve this issue.

10. Conclusions and Future Work

This paper proposes two traffic control system techniques based on replication and diversity (N-versions), which substantially increase resilience to different kinds of attacks or natural accidents. These two techniques were studied mathematically and shown to drastically reduce the probability of failures and, consequently, the corresponding possible damage. With technique1, we can see that the optimal results were obtained when the number of versions was 3, as explained earlier. With technique2, we can see that one should implement any odd number of versions up to and beyond 3 to obtain a resilient system, depending on the cost of possible damages caused by any failure, as explained earlier. We are convinced that the only drawback of the proposed techniques is their cost, since they use replication and diversity (N-versions); however, we are also convinced that resilience cannot be reached without these two last techniques, as explained and proved earlier. We think that studying ways to reduce the time overhead of voters in both systems and acceptance tests in the second system could be a priority in future work.

Author Contributions: Conceptualization, A.B., M.K., F.E.E., J.M.Q. and K.J.; methodology, A.B., M.K., F.E.E., J.M.Q. and K.J.; software, J.M.Q., A.B. and M.K.; validation, J.M.Q., M.K. and A.B.; investigation, A.B., M.K., F.E.E., J.M.Q. and K.J.; writing—original draft preparation, A.B., M.K., F.E.E., J.M.Q. and K.J.; writing—review and editing, J.M.Q. and M.K.; visualization, J.M.Q., A.B., M.K., F.E.E. and K.J.; supervision, F.E.E. and K.J.; project administration, A.B., M.K., F.E.E. and K.J.; funding acquisition, A.B. All authors have read and agreed to the published version of the manuscript.

Funding: This project was funded by the Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah, under grant no. (RG-8-611-40). The authors, therefore, acknowledge, with thanks, the technical and financial support of DSR.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Goldman, H.; McQuaid, R.; Picciotto, J. Cyber resilience for mission assurance. In Proceedings of the 2011 IEEE International Conference on Technologies for Homeland Security (HST), Boston, MA, USA, 15–17 November 2011; pp. 236–241.
2. Ali, Q.; Ahmad, N.; Malik, A.; Ali, G.; Rehman, W. Issues, Challenges, and Research Opportunities in Intelligent Transport System for Security and Privacy. *Appl. Sci.* **2018**, *8*, 1964. [CrossRef]
3. Perrine, K.; Levin, M.; Yahia, C.; Duell, M.; Boyles, S. Implications of Traffic Signal Cybersecurity on Potential Deliberate Traffic Disruptions. *Transp. Res. Part A Policy Pract.* **2019**, *120*, 58–70. [CrossRef]
4. Wang, X.; Miao, S.; Tang, J. Vulnerability and Resilience Analysis of the Air Traffic Control Sector Network in China. *Sustainability* **2020**, *12*, 3749. [CrossRef]
5. Comert, G.; Pollard, J.; Nicol, D.; Palani, K.; Vignesh, B. Modeling Cyber Attacks at Intelligent Traffic Signals. *Transp. Res. Rec. J. Transp. Res. Board* **2018**, *2672*, 76–89. [CrossRef]
6. Miller, C.; Valasek, C. Remote Exploitation of an Unaltered Passenger Vehicle. *Defcon* **2015**, *23*, 1–91.
7. Mendes, D.; Ivaki, N.; Madeira, H. Effects of GPS spoofing on unmanned aerial vehicles. In Proceedings of the 2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC), Taipei, Taiwan, 4–7 December 2018; pp. 155–160.
8. Ganin, A.; Mersky, A.; Jin, A.; Kitsak, M.; Keisler, J.; Linkov, I. Resilience in Intelligent Transportation Systems (ITS). *Transp. Res. Part Emerg. Technol.* **2019**, *100*, 318–329. [CrossRef]
9. Klingensmith, K.; Madni, A. Resilience Concepts for Architecting an Autonomous Military Vehicle System-of-Systems. In *Disciplinary Convergence in Systems Engineering Research*; Springer: Cham, Switzerland, 2017; pp. 65–81. [CrossRef]
10. Linkov, I.; Kott, A. Fundamental concepts of cyber resilience: Introduction and overview. In *Cyber Resilience of Systems and Networks. Risk, Systems and Decisions*; Springer: Cham, Switzerland, 2019; pp. 1–25.
11. Avizienis, A.; Laprie, J.; Randell, B.; Landwehr, C. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable Secur. Comput.* **2004**, *1*, 11–33. [CrossRef]
12. Clark, K.; Bhatia, U.; Kodra, E.; Ganguly, A. Resilience of the U.S. National Airspace System Airport Network. *IEEE Trans. Intell. Transp. Syst.* **2018**, *19*, 3785–3794. [CrossRef]
13. Langford, M.; Simon, G.; McKinley, P.; Cheng, B. Applying Evolution and Novelty Search to Enhance the Resilience of Autonomous Systems. In Proceedings of the 2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), Montreal, QC, Canada, 25–26 May 2019; pp. 63–69.
14. Bianchin, G.; Pasqualetti, F.; Kundu, S. Resilience of Traffic Networks with Partially Controlled Routing. In Proceedings of the 2019 American Control Conference (ACC), Philadelphia, PA, USA, 10–12 July 2019; pp. 2670–2675;
15. Khalghani, M.; Verma, V.; Khushalani Solanki, S.; Solanki, J. Resilient Networked Control of Inverter-Based Microgrids against False Data Injections. *Electronics* **2022**, *11*, 780. [CrossRef]
16. Cho, B.; Kim, S.; Kim, K.; Park, K. A Controller Switching Mechanism for Resilient Wireless Sensor—Actuator Networks. *Appl. Sci.* **2022**, *12*, 1841. [CrossRef]
17. Abudayyeh, D.; Nicholson, A.; Ngoduy, D. Traffic Signal Optimisation in Disrupted Networks, to Improve Resilience and Sustainability. *Travel Behav. Soc.* **2021**, *22*, 117–128. [CrossRef]