Consistency Before Availability: Network Reference Point based Failure Detection for Controller Redundancy

Bjarne Johansson*, Mats Rågberger*, Alessandro V. Papadopoulos†, Thomas Nolte†
* ABB Process Automation, Process Control Platform, Västerås, Sweden, {bjarne.johansson,mats.ragberger}@se.abb.com
† Mälardalen University, Västerås, Sweden, {alessandro.papadopoulos, thomas.nolte}@mdu.se

***Abstract*—Distributed control systems constitute the automation solution backbone in domains where downtime is costly. Redundancy reduces the risk of faults leading to unplanned downtime. The Industry 4.0 appetite to utilize the device-to-cloud continuum increases the interest in network-based hardware-agnostic controller software. Functionality, such as controller redundancy, must adhere to the new ground rules of pure network dependency. In a standby controller redundancy, only one controller is the active primary. When the primary fails, the backup takes over. A typical network-based failure detection uses a cyclic message with a known interval, a.k.a. a heartbeat. Such a failure detection interprets heartbeat absences as a failure of the supervisee; consequently, a network partitioning could be indistinguishable from a node failure. Hence, in a network partitioning situation, a conventional heartbeat-based failure detection causes more than one active controller in the redundancy set, resulting in inconsistent outputs. We present a failure detection algorithm that uses network reference points to prevent network partitioning from leading to dual primary controllers. In other words, a failure detection that prioritizes consistency before availability.**

I. INTRODUCTION

Distributed control systems (DCSs) progress toward more network-oriented architectures where switched Ethernet in combination with OPC UA[1] constitute the interoperability communication backbone in future automation installations [1]. A progression observable through the increase of Ethernet solutions and decrease of fieldbus installations [2], [3].

The trajectory to network-centric architectures yields that DCS controllers, denoted Distributed Control Nodes (DCN) by the Open Process Automation Forum (OPAF), and Fieldbus Communication Interfaces (FCI) will rely more on Ethernet. Ethernet enables new controller deployment alternatives, such as the execution of control applications in a virtualized context in the cloud or in orchestrated embedded clusters [4], [5], [6].

DCN redundancy is an example of functionality that must refrain from customized hardware dependency to avoid reducing deployment alternatives. Today, standby redundancy with hardware duplication is a typical redundancy pattern in a DCS context. One DCN is the active primary DCN, and the other is the passive backup DCN, ready to take the primary role in case of failure of the current primary. Only the primary DCN provides output values to I/O connected to the process and physical world.

A common failure detection method is a heartbeat, a cyclic message sent within a known interval from the supervisee to the supervising [7]. The supervising node interprets heartbeat timeout as a supervisee failure. We denote such a conventional heartbeat-based failure detection Conv. FD.

In a DCN redundancy context, the supervisee is the primary DCN, and the backup DCN is the supervising. The backup DCN in a DCN redundancy pair using Conv. FD interprets a heartbeat timeout as a primary DCN failure and resumes the primary role. However, a timeout is not necessarily a consequence of a primary DCN failure; it could follow from a network failure causing a network partitioning between the primary and backup DCN. Hence, in a network partitioning situation, with a DCN redundancy using Conv. FD, the partitioning causes the backup DCN to become primary while the former primary remains primary, resulting in dual primaries. A consequence of dual primaries is that both the DCNs in the DCN redundancy pair control I/O values, causing inconsistency.

DCN redundancy is typically used with network redundancy to reduce the probability of communication failures. However, the partitioning probability is not zero, not even with duplicated networks. Hence, DCN redundancy based on Conv. FD gives a dual primary probability larger than zero.

If the DCN redundancy cannot ensure a single primary, it is better to have no output than conflicting outputs from dual primaries. This prevents fluctuations in the controlled process because I/O channels that expect DCN updated values will output preconfigured values when updates are missing. This setting is named Output Set as Predetermined (OSP) [8] in ABB I/O system context, and the input channel equivalent is Input Set as Predetermined (ISP), which the DCN control application uses when input values are absent.

Fig. 1 shows a partitioning situation, with two network failures $F1$ and $F2$ between the redundant DCN pair. Both the DCNs (DCN A1 and DCN A2) take the primary role when using Conv. FD and provide output values in the resulting dual primary situation. FCI B gets output values from both DCNs. Values that differ since the DCNs reach two different FCIs.

This work is funded by The Knowledge Foundation (KKS), project ARRAY, and by The Swedish Foundation for Strategic Research (SSF), project FuturAS.

[1]https://opcfoundation.org/

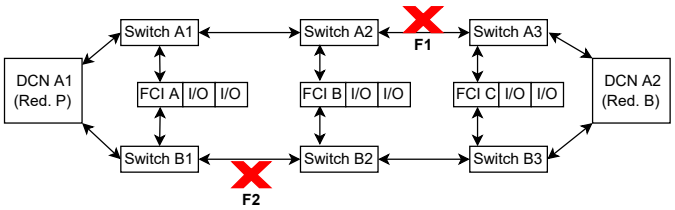

Fig. 1. Example of a network failure causing a partitioning where both DCN will become primary and drive output signals. The DCN redundancy recovery time is typically between ten to hundred milliseconds.

DCN A1 reaches FCI A and B, while DCN A2 reaches FCI B and C. Hence, the two DCNs will use different input values. DCN A1 will use the ISP values instead of actual I/O values from FCI C, while DCN A2 will use the actual I/O values from FCI C. Vice versa applies to FCI A and the DCNs.

We address this problem by proposing a failure detection algorithm that utilizes a Network Reference Point (NRP) external to the DCNs to mitigate dual primary situations due to network partitioning while striving to keep high availability. We use the name NRP Failure Detection (NRP FD) for the proposed algorithm.

The paper is structured as follows, Section II presents the related work, and Section III describes NRP FD. Section IV compares NRP FD and Conv. FD concerning the availability and consistency tradeoff, followed by an experimental comparison of NRP FD and Conv. FD described in Section V. Lastly, Section VI summarizes the paper.

II. Related work

The Consistency, Availability, and Partition tolerance (CAP) theorem [9], [10] state that in case of partitioning, a distributed system can be either consistent or available, not both. Lee et al. present a modified version of CAP, the Consistency, Availability, and apparent Latency (CAL) theorem [11], [12]. Using CAL, Lee et al. quantify consistency and availability compromises under latency requirements. NRP FD preserves consistency under network partitioning by ensuring one or no primary, i.e., availability before consistency, further described in Section III.

Failure detection is crucial in a standby redundancy solution, and existing work ranges from the introduction of unreliable failure detection concept [13] and failure detection Quality of Service (QoS) attributes [14] to heartbeat optimizations and improvements [15], [16]. However, none of these works addresses differentiation between node and network failure, which, as pointed out by van Steen [17], a failure detector ideally should do. Distinguishing node and network failures could solve the problem we address.

We have not found any scientific failure detection publication addressing the differentiation between node and network failures in wired networks, but two patents that do. Charny et al. [18] uses an alternative path to the node when failing to reach the node on the first path, a solution that is similar to the neighboring using approach van Steen [17] discuss. I.e., querying neighbors of the suspected node to learn if they can reach it. Filsfils et al. [19] describe a Bidirectional Forwarding Detection (BFD) based approach to distinguish network from node failures using multiple BFD sessions over various disjoint paths. BFD [20] is a protocol for quick link failure detection between adjacent nodes, typically used by routers.

Ritter et al. [21] present a similar approach for ad-hoc mobile wireless networks. A beacon node sends heartbeats to all other nodes. The beacon's closest neighbor, the buddy node, supervises the beacon faster than the heartbeat interval. If the buddy node does not hear the beacon, it tries an alternative path, and if that also fails, the beacon node is assumed to have failed. Otherwise, the network is considered partitioned. Our work addresses partitioning without alternative paths. The work described above would treat such partitioning as a node failure and cause a dual primary situation, which we want to avoid.

Failure detection in a redundant DCN context, with only one backup, constitutes an implicit leader election. The Bully algorithm [22] is one of the more famous leader election algorithms, and many variants exist [23], [24], [25]. However, they will all elect a leader per partition, i.e., provide availability before consistency.

Quorum consensus protocols like Paxos [26], [27] and Raft [28] provide consistency and tolerates $(N-1)/2$ faults, where N is the number of nodes. A redundant DCN only requires two individual DCNs, $N = 2$, i.e., a consensus protocol-based redundancy would not tolerate one fault if $N = 2$, making quorum-based DCN redundancy meaningless.

Active redundancy means that all the nodes in the redundant set are active [29], [30]. However, it pushes the decision of which data to use to a selection function such as a voter, a selection that needs to be made on the data consumers to be partition tolerant, forcing DCN redundancy data handling to all data consumers.

None of the related work covered solves the problem we address, a real-time capable failure detection that prioritizes consistency without requiring a DCN quorum while minimizing the availability compromise.

III. Network reference point failure detection

A. Overview

NRP FD is a heartbeat-based failure detection with additions. We describe it in the context of a redundant DCN pair. A failed DCN is assumed to stop sending heartbeats. Generalization and adaptation to more flexible redundancy patterns are future work.

NRP FD provides consistency over availability, meaning that both DCNs will not be primary due to multiple network failures, but none might.

The NRP FD additions are (i) the usage of the NRP and (ii) the optional utilization of temporal properties of received heartbeats. We use the word network to describe a communication path between primary and backup DCN. NRP FD does not dictate any requirement on the number of networks connecting the primary and backup. In the description, we use a redundant

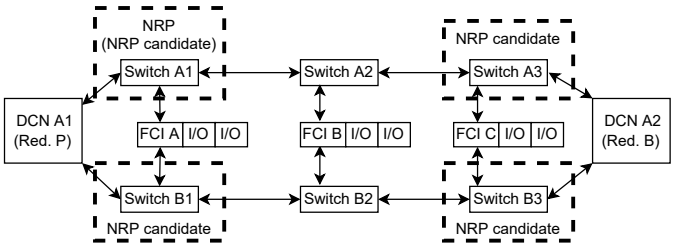

Fig. 2. NRP in a system context where each DCN has one candidate per network and the primary has appointed one NRP (switch A1).

network (two networks), exemplified in Fig. 2. NRP FD assumes that the NRP is an individually accessible node in the network infrastructure between primary and backup; in practice, a managed switch. Each DCN in the redundant pair has an NRP candidate per network; see Fig. 2. How NRP FD is made aware of the NRP candidates is outside the algorithm's scope. Section III-B describes the considerations that apply to the NRP candidates. Only one of the NRP candidates is the NRP; selected by the primary.

Algorithm summary; the primary sends heartbeats on all networks connecting the primary and the backup DCNs, containing the IP address of the NRP that the primary has selected. If the backup does not observe any heartbeat within a timeout period, it checks if it can reach the NRP, and if it can, it becomes the new primary, and if it cannot, it remains passive.

We use the term PINGNRP for the NRP reachability test. NRP FD is agnostic to the implementation behind PINGNRP. To comply with COTS switches, the PINGNRP is limited to communication means and protocols supported by most managed industrial COTS switches; in practice, this means using ICMP[2] ping.

An ICMP ping does not have a bounded response time, but NRP FD can guarantee a bounded reaction time by utilizing temporal aspects of heartbeats received on the different networks. NRP FD can assume that a heartbeat silence is due to a primary failure if heartbeats timeout simultaneously on multiple networks and skip the PINGNRP. Note that this is optional handling with the cost of a small risk of treating simultaneous network failures as a failure of the primary.

A backup that does not use the simultaneous timeout optimization or only has one functioning network always uses the PINGNRP to determine if it should become the primary in case of heartbeat silence.

In addition to the PINGNRP performed by the backup, NRP FD prescribes that the primary checks if it can reach the NRP; if it can't, it tries to elect a new NRP from the NRP candidates. If that fails, it surrenders the primary role.

If a backup does not receive heartbeats on the network of the NRP but on other networks, the backup checks if the NRP is reachable. If it is not, the backup proposes a switch of NRP to the primary.

[2]https://www.rfc-editor.org/rfc/rfc792

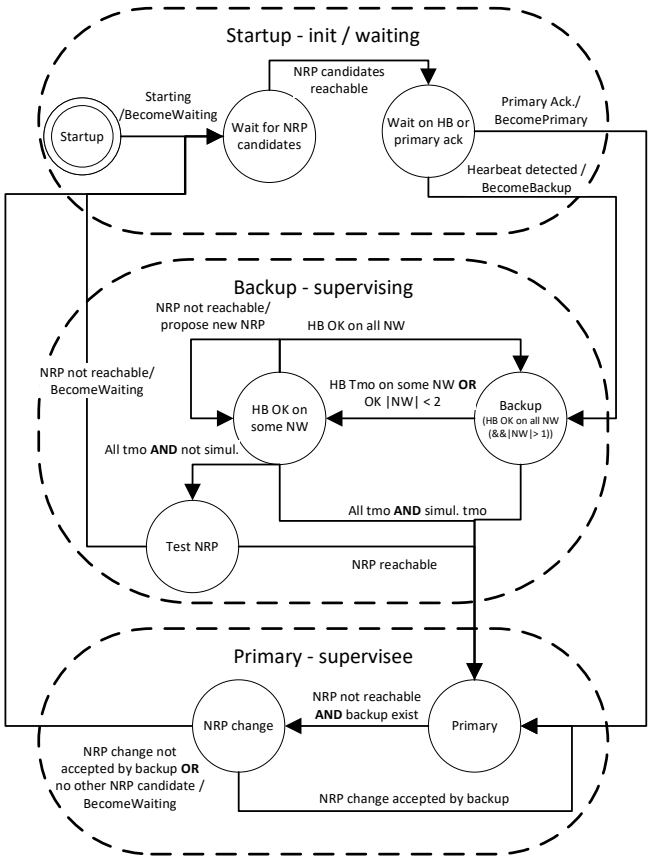

Fig. 3. NRP FD state machine.

Fig. 3 gives an overview of NRP FD, further described in Section III-C.

B. Assumptions

NRP FD requires that the NRP (i) has no common cause failure with the DCN and (ii) that the NRP is uniquely addressable.

The NRP must not have any common cause failure with the DCN, such as being powered by the same power supply or using the same CPU for processing requests, such as ICMP pings. That could be the case if the NRP were an embedded switch mounted on the DCN hardware.

The DCN is assumed to have an IP address per network, and the IP addresses of the partner DCN per network, are known for each DCN. The NRP candidates' IP addresses are preconfigured and unique, making the NRP candidates uniquely addressable.

C. Detailed description

We begin the detailed description with the startup and the transition to the specific roles, followed by NRP selection. Then, we continue with the actual failure detection, the (i) backup - supervision handling, and the (ii) primary - supervisee handling. Finally, the NRP change handling wraps up the section.

1) Startup - init / waiting: Algorithm 1 describe the startup and initial handling. It consists of the function BECOMEWAITING that does three things (i) wait for available NRP candidates, (ii) wait for acknowledgment to become primary or a heartbeat from an existing primary, (iii) transition to the primary or backup role.

WAITFORNRPCANDIDATES waits for at least one NRP candidate to be available. Available means answering to PINGNRP. When at least one NRP candidate is available, the next step is to wait for either a heartbeat from an existing primary or an OK-to-be-primary acknowledgment. The function HBORPRIMARYACK performs this wait. An operator or maintenance engineer issues the primary acknowledgment, okaying the DCN to become primary, mitigating the risk that a DCN starting up in a partitioned network becomes primary. The person issuing the acknowledgment must ensure that this is not the case.

Algorithm 1 Startup - init / waiting

```
1:  BECOMEWAITING( )
2:  function BECOMEWAITING( )
3:      WAITFORNRPCANDIDATES( )
4:      do
5:          do
6:              hbOrAckSts ← HBORPRIMARYACK( )
7:          while hbOrAckSts ≠ HbOrPrimaryAck
8:          if hbOrAckSts = AckToBePrimary then
9:              BECOMEPRIMARY( )
10:         else                          ▷ HB seen, become backup
11:             BECOMEBACKUP( )
12:         end if
13:     while not(isPrimary OR isBackup)
14: end function
15: function WAITFORNRPCANDIDATES( )
16:     do
17:         MONITORNRP( )                 ▷ See Algorithm 2
18:     while reachableCandidates = {∅}
19: end function
20: function BECOMEPRIMARY
21:     SELECTNRP( )                      ▷ Select the NRP to use.
22:     isPrimary ← TRUE
23: end function
24: function BECOMEBACKUP
25:     SENDIMHERETOPRIMARY( )
26:     isOwnIpInHb ←WAITFORONWADDRINHB(Tmo)
27:     if isOwnIpInHb == TRUE then
28:         isBackup ← TRUE
29:     else
30:         isBackup ← FALSE
31:     end if
32: end function
```

2) BecomePrimary - transition to the primary role: The transition to the primary role only consists of selecting an NRP from the NRP candidates ($reachableCandidates$ from Algorithm 2), Line 20-23 in Algorithm 1. Any NRP candidate reachable is selectable.

3) BecomeBackup - transition to the backup role: The backup informs the primary of its presence, Line 25-27 in Algorithm 1. Then, the backup waits for the primary to acknowledge its presence by adding the IP address of the backup to the heartbeat field *Backups Known*, see Table I. This ensures that the backup does not resume the primary role due to a failure when the primary is unaware of the backup. Since a primary without backups will remain primary even if the NRP reachability is lost. A backup is not a backup unless it sees its address in the heartbeat field *Backups Known*, continuously checked while in the backup state.

4) NRP selection and candidate monitoring: The set $reachableCandidates$, contains the reachable NRP candidates; see Algorithm 2. The primary selects an NRP from the $reachableCandidates$ set as the NRP. The $reachableCandidates$ set is updated with a suitable interval, for example, a few times per minute. Each network has an NRP candidate. The algorithm description does not cover how NRP FD becomes aware of NRP candidates; however, Section V-A presents alternatives. The function GETCANDFORNW represents NRP candidate retrieval for a specific network nw.

Algorithm 2 NRP candidate monitoring

```
1:  function MONITORNRP( )
2:      reachableCandidates ← {∅}
3:      for all nw ∈ AllNetworks do
4:          candidate ← GETCANDFORNW(nw)
5:          isReachable ← PINGNRP(candidate)
6:          if isReachable then
7:              reachableCandidates ← reachableCandidates ∪ {candidate}
8:          end if
9:      end for
10: end function
```

5) Backup - supervising: The backup continuously, with a cycle time preferably longer than the heartbeat interval, announces its presence to the primary, see Algorithm 3 Line 2. Further described in Section III-C6.

The backup checks the heartbeats on all used networks, Line 3. If heartbeats timed out on all networks and two or more timed out simultaneously, NRP FD assumes that the cause is a failure of the primary rather than two (or more) independent network failures in a short time frame, Line 3-5. Treating a simultaneous timeout of heartbeats as a primary failure is an optimization, a way to reduce decision time by avoiding PINGNRP when a failover time shorter than the response time of the PINGNRP is required. If the PINGNRP have a sufficiently low upper bounded reply time, Line 3-5 could be removed and be covered by the handling described on Line 6-10.

Equation 1 defines the simultaneous timeout, where $\Delta hbTmo$ is the interval for considering two heartbeats to be simultaneous, $hbTmoT$ is the time of the timeout, and NW

Algorithm 3 Backup - supervising

```
1: while isBackup do
2:     SendImHereToPrimary( )        ▷ Longer interval.
3:     hbSts = ChkHbStsOnAllNw( )
4:     if hbSts == tmoAllSimul then
5:         BecomePrimary( )
6:     else if hbSts == tmoAllNotSimul then
7:         isNrpReachable = PingNRP( )
8:         if isNrpReachable then
9:             BecomePrimary( )
10:        end if
11:    else if hbSts == tmoSomeNotAll then
12:        isNrpReachable = PingNRP( )
13:        if not isNrpReachable then
14:            AskPrimaryToSwitchNRP( )
15:        end if
16:    end if
17:    isBackup ← isOwnIpInHb        ▷ See Alg. 1
18: end while
```

Algorithm 4 Primary - supervisee

```
1: nrpAddr ← GetNrpAddr( )
2: while isPrimary do
3:     SendHeartbeat(nrpAddr)
4:     isNRPReachable ← PingNRP(nrpAddr)
5:     if not isNRPReachable then
6:         newNRPAddr ← GetNewReachableNrp( )
7:         if IsValid(newNRPAddr) then
8:             if isBupAvailable then
9:                 AskBupToChgNrp(newNRPAddr)
10:                isNRPChgOk ← NrpChgResp(Tmo)
11:                if isNRPChgOk then
12:                    nrpAddr ← newNRPAddr
13:                else
14:                    BecomeWaiting( )      ▷ See Alg. 1
15:                end if
16:            else
17:                nrpAddr ← newNRPAddr
18:            end if
19:        else
20:            if isBupAvailable then
21:                BecomeWaiting( )          ▷ See Alg. 1
22:            end if
23:        end if
24:    end if
25:    isBupAvailable ← IsBackupPresent( )
26: end while
```

is the set of all networks connecting the primary and backup. $hbTmoT$ is zero in case of no timeout.

$$i, j \in NW, \forall hbTmoT_i, \forall hbTmoT_j, \\ i \neq j, hbTmoT_i \neq 0, hbTmoT_j \neq 0 \mid \\ |hbTmoT_i - hbTmoT_j| \leq \Delta hbTmo \tag{1}$$

If heartbeats timeout on all networks but not simultaneously (or if the optimization is not used), the NRP reachability test determines if the backup should become the primary, Line 6-10.

If heartbeats timed out on some, but not all, networks, the backup tests the NRP reachability. If unreachable, the backup asks the primary to change NRP, Line 11-16; further described in Section III-C7.

6) Primary - supervisee: The primary sends heartbeats on all networks, with the configured interval containing the address of the current NRP; see Algorithm 4 Line 3.

The primary also checks the NRP reachability, Line 4. A reachable NRP does not require any further actions. The remaining parts of Algorithm 4 pseudocode cover the unreachable NRP scenario, Line 5-24. If a backup and an alternative NRP candidate exist, the primary tries to switch NRP, Line 7-18.

The primary must ensure that the backup accepts an NRP change before changing NRP. Hence, the primary requests the backup to change NRP and waits for a response with a timeout, Line 9-15. The primary leaves the primary role if it does not receive a positive confirmation in time. If no backup exists, the primary just switches NRP.

If no reachable NRP candidate exists, Line 19-23, the primary leaves the primary role if the backup is present and remains primary if the backup is not present. The primary expects the backup to report its presence within a timeout, see Line 25, allowing the primary to vacate the primary role only if a backup is present. The backup presence timeout is preferably at least one order of magnitude larger than the heartbeat interval since backup presence detection does not affect the failover time.

7) NRP change handling: The backup and primary must never use different NRPs; an NRP change must never violate that. It is beneficial to change the NRP in two situations. The first is when the primary fails to reach the NRP. In that situation, the primary proposes a new NRP to the backup as shown in Algorithm 4. A backup that accepts the new NRP starts using it after it sees the new NRP address in a heartbeat. Hence, during the time between acceptance and heartbeat receiving, the backup can not use the NRP reachability to become primary. The backup can only know that the acceptance message is delivered to the primary once it sees a confirmation. The confirmation is the changed NRP address in received heartbeats. If the NRP address is not changed, the backup reverts to using the former NRP address after two heartbeat cycles by using the iteration number in the heartbeat message, see Table I.

The second situation is when the backup cannot reach the NRP. A backup that cannot reach the NRP cannot take the primary role using the NRP reachability test. Hence, in that situation, the backup suggests that the primary switches the NRP.

TABLE I
HEARTBEAT MESSAGE FIELDS.

Name	Description
NRP	Address of the current NRP
Backups Known	The addresses of the backups known by the primary.
Iteration (seq.) number	Identifies the iteration/cycle the the heartbeat was sent. Incremented each cycle by the primary.

IV. CONSISTENCY AND AVAILABILITY COMPARISON

In this section, we present a consistency and availability comparison between Conv. FD and NRP FD using switch failure scenarios. We use two topologies, $T1Sw$ with one switch per network between the DCN redundancy pair and $T3Sw$ with three switches, depicted in Fig. 2.

We assume that all switches, and thereby the NRP and NRP candidates, are the same and have the same MTTF. We use a switch MTTF of 75 years, same as the Westermo managed-switch Lynx[3]. The DCN MTTF is assumed to be lower since DCNs are likely more complex hardware vise. We assume a DCN MTBF of 20 years.

The reliability function $R(t) = e^{-\lambda t}$ and the corresponding failure function $F(t) = 1 - R(t)$ give the failure probabilities. We use a run time of ten years, i.e., $t = 10$. The scenarios are multiple faults, requiring the failure of one network path between the primary and backup and another failure. One week is the assumed reparation time for the first failure. Heartbeats are sent from the primary to the backup every 50 milliseconds, Line 3 in Algorithm 4, and the backup presence timeout $IsBackupPresent$, Line 25, is one second. Table II presents the probabilities for the scenarios described below.

The scenarios where NRP FD prioritizes consistency before availability by vacating the primary or backup role potentially negatively impact availability. Therefore, these are the scenarios used for comparison. In other words, we compare the probability of scenarios that lead to no primary using NRP FD with the likelihood of scenarios leading to dual primary using Conv. FD.

NRP FD vacates the primary role if the NRP is not reachable, a backup is present, and the request to change NRP fails, see Line 5-24 in Algorithm 4. The NRP change request can fail for two reasons, partitioning or backup failure. In the partitioning case, the backup will become primary unless the second failure is a failure of the NRP itself, denoted $NRPFdNRPAndNWFail$, failure combination $F1F4$ in Fig.4. For $NRPFdNRPAndNWFail$ to result in a no primary situation, the NRP must fail while the other network path is broken. Hence the probability increases with the reparation time. The second reason and scenario for an NRP change request to fail is a backup failure simultaneously to the primary loss of NRP reachability, denoted $NRPFdNRPAndBupFail$.

[3]https://www.westermo.se/-/media/Files/Data-sheets/westermo_ds_lynx_100-and-200-series_2205_en_revg.pdf

TABLE II
AVAILABILITY AND CONSISTENCY LOSS PROBABILITIES.

Scenario	Probability	
	NRP FD No primary	Conv. FD Dual primary
$NRPFdNRPAndNWFail$	$T1Sw$: $3.2 * 10^{-3}\%$ $T3Sw$: $8.5 * 10^{-3}\%$	0% [1]
$NRPFdNRPAndBupFail$	$T1Sw$: $1.7 * 10^{-8}\%$ $T3Sw$: $1.7 * 10^{-8}\%$	0%
$NRPFdBupNotRchNRP$	$T1Sw$: $2.0 * 10^{-9}\%$ $T3Sw$: $5.2 * 10^{-9}\%$	0%
$ConvFdNwPart$	See note[2]	$T1Sw$: $3.2 * 10^{-3}\%$ $T3Sw$: $2.5 * 10^{-2}\%$

[1] Partitioning due to NRP failure is included in $ConvFdNwPart$ for Conv. FD.

[2] Partitioning due to failure of any switch is covered by $NRPFdNRPAndNWFail$ for NRP FD.

For $NRPFdNRPAndBupFail$ to cause a non-primary situation, the NRP must fail before the $IsBackupPresent$ expires, see Line 25 in Algorithm 4.

If the backup cannot reach the NRP, it cannot resume the primary role using the PINGNRP. Therefore, it proposes an NRP change; see Line 12-15 in Algorithm 3. If the primary fails before it has changed the NRP, the backup won't takeover, denoted $NRPFdBupNotRchNRP$. We use 100 milliseconds as the NRP change time, i.e., twice the heartbeat interval.

Conv. FD causes a dual primary situation in any network partitioning scenario, denoted $ConvFdNwPart$.

Table II shows that the inconsistency probability for Conv. FD is higher than NRP FD's availability loss probability for topology $T3Sw$. The main reason for that is that a Conv. FD causes a dual primary situation for any partitioning situation, i.e., if any of the switches between the primary and backup fail on the two networks. NRP FD only causes a no primary situation when the specific failure mentioned above occurs. For $T1Sw$, the probability of availability loss with NRP FD is marginally higher than the probability for consistency loss using Conv. FD.

V. IMPLEMENTATION AND EVALUATION

A. Implementation

The NRP candidates are the switches, see Table III and Fig. 4, that NRP FD learns about using a dynamic approach based on Link Layer Discovery Protocol (LLDP). LLDP allows neighboring devices to announce their presence. The switches reveal their existence and IP addresses with LLDP, and NRP FD in each DCN learns about adjacent switches on each network. Those adjacent switches are the NRP candidates, from which NRP FD selects an NRP.

LLDP provides vendor-specific extension possibilities. Hence, LLDP could announce the support of low latency PINGNRP that NRP FD could utilize if available. For example, a future switch supporting a real-time PINGNRP could reveal that through LLDP, and NRP FD could use this knowledge to avoid using simultaneous

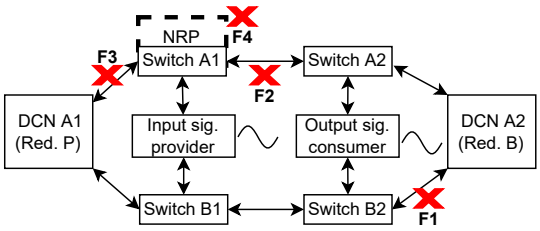

Fig. 4. The evaluation setup used. F1 - F4 marks the network faults induced.

TABLE III
SOFTWARE AND HARDWARE USED.

Node name	Hardware	Software
DCN A1	MSI, Core i3	VxWorks 7.0
DCN A2	Lenovo, Core i5	VxWorks 7.0
Input sig. provider	Intel NUC, Core i3	VxWorks 7.0
Output sig. consumer	Lenovo, Core i7	Windows 10
Switches	Zyxel, GS1900-8	V2.40

heartbeat timeout handling when switches that support real-time capable PINGNRP are used.

Our implementation uses ICMP ping as the PINGNRP, and our test shows that our switches typically reply to ping with a sub-millisecond response time.

B. Evaluation

Fig. 4 shows the evaluation setup we use. Input sig. provider emulates an input I/O. It generates a sinus wave with a frequency of two Hertz and an update rate of 100 updates per second. Outputting a new value every ten milliseconds on network A and B, using UDP multicast.

The control application that runs on the redundant DCN pair (DCN A1 and DCN A2) expects Input sig. provider values, and outputs values to the Output sig. consumer. The control application checks for an updated value every five ms. If received, it outputs the value received; if it does not receive a value after three iterations, it outputs the ISP value zero.

The Output sig. consumer emulates an output I/O and uses the last received value as OSP. It is a Windows application that plots the received value for visualization purposes; Fig. 6 shows the Output sig. consumer value plot when OK.

We compare our NRP FD and Conv. FD implementation and show the resulting consistency difference between NRP FD and Conv. FD in network partitioning situations by breaking the network in the places marked $F1$ - $F4$ in Fig. 4. Fault $F1$ breaks network B and is always the first fault. We combine $F1$ with $F2$ - $F4$ separately, resulting in three different fault combinations, (i) $F1F2$, (ii) $F1F3$, and (iii) $F1F4$.

Fig. 5 shows the Output sig. consumer plot with fault combination $F1F2$ using Conv. FD. DCN A1 receives values from Input sig. provider while DCN A2 does not. Hence, DCN A1 outputs the sine wave values, and DCN A2 uses the ISP value and outputs a constant zero. Output sig. consumer receives values from both DCN A1 and DCN A2, resulting in the distorted sine wave shown in Fig. 5. With the same fault combination, $F1F2$, NRP FD ensures that DCN A1 remains primary and DCN A2 passive. Hence, the Output sig. consumer receives consistent values, shown in Fig. 6.

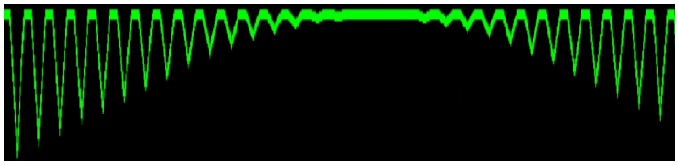

Fig. 5. Conv. FD based redundancy causes dual primaries in partitioning situation $F1F2$. Output sig. consumer receives different (inconsistent) values from DCN A1 (over the lower network) and DCN A2 (over the uper network).

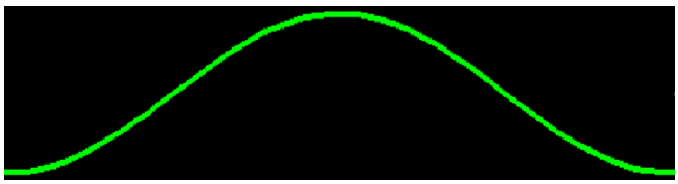

Fig. 6. Correctly received and plotted values by Output sig. consumer. NRP FD ensures one primary; hence, Output sig. consumer receives the correct sine wave values even under partitioning.

The fault combination $F1F3$ results in a dual primary situation using Conv. FD. The difference from $F1F2$ is that DCN A1 and DCN A2 receive values from the Input sig. provider. Hence, the inconsistency in the output values is less than for $F1F2$, as shown in Fig. 7, but it still shows. With NRP FD, DCN A2 becomes primary, and DCN A1 vacates the primary role, keeping consistency as shown in Fig. 6.

Fault combination $F1F4$ covers NRP failure ($F4$) and the Conv. FD result is the same as for $F1F2$, shown in Fig. 5. Fault combination $F1F4$ using NRP FD results in DCN A1 vacating the primary role; it cannot reach the NRP nor elect a new NRP since it cannot communicate with DCN A2. DCN A2 does not become primary since it can't reach the failed NRP. Hence, no primary and Output sig. consumer, do not get any values and therefore use the OSP value, shown in Fig. 8.

Performance is also an important aspect, and the penalty using NRP FD comes from the additional PINGNRP, realized with ICMP ping in our evaluation. However, the overhead is typically less than a millisecond. We used a heartbeat period

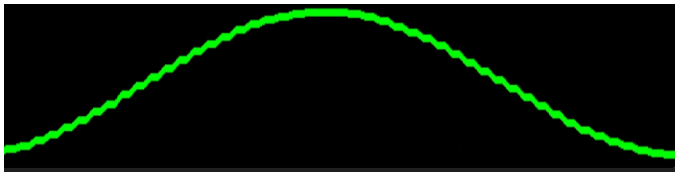

Fig. 7. Conv. FD caused dual primary, where both DCN use the same input values, hence inconsistency is less than for $F1F2$, shown in Fig. 5, but still visible when compared to Fig. 6.

Fig. 8. No value received - Output sig. consumer plots the last value, as OSP.

of five milliseconds and required two missing heartbeats to indicate a failure. With that configuration, we did not see any difference when measuring the time between two consecutive updates in the Output sig. consumer when primary failed.

More elaborated performance measurements are future work. Future implementation optimized for performance and combined with a time bounded low latency implementation in the switches for the PINGNRP.

VI. SUMMARY AND FUTURE WORK

This paper presents NRP FD, a failure detection algorithm that prioritizes consistency, in contrast to the conventional failure detection, Conv. FD, that prioritizes availability. The target use case, and the description base of the NRP FD algorithm, is controller redundancy. With a theoretical comparison between NRP FD and Conv. FD we showed that the NRP FD is less likely to lose availability than Conv. FD is to lose consistency. Further, with a testbed and implementations of NRP FD and Conv FD, we showed the consistency gain NRP FD gives over Conv. FD by injecting failures causing network partitioning between the redundant pair.

Potential continuations include incorporating a hard real-time, low-latency PINGNRP support in the switches to perform a more exhaustive and challenging failover performance test. With tailored switch support, the heartbeat and PINGNRP could be integrated to allow NRP FD to guarantee at most one primary, not only under persistent network partitioning but also under transient disturbance. Future work would aim to prove that property using model checkers and probabilistic network models.

Furthermore, a low-latency PINGNRP can also be used for performant network supervision and breakage localization if combined with a topology map. Additional future work is to combine NRP FD with failure detection and role selection targeting other redundancy configurations than 1oo2, such as Heartbeat bully [31].

REFERENCES

[1] "The dcs of tomorrow - abb's process automation system vision whitepaper." https://new.abb.com/control-systems/control-systems/envisioning-the-future-of-process-automation-systems/automation-system-whitepaper. Accessed: 2023-03-20.

[2] P. Ganesan, S. Gunasekaran, A. A. Balaji, S. J. Fathima, and S. Suryaprakash, "Comparative analysis on industrial iot communication protocols and its future directives," in *2021 International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA)*, pp. 1–6, IEEE, 2021.

[3] G. Soós, D. Ficzere, and P. Varga, "Investigating the network traffic of industry 4.0 applications–methodology and initial results," in *2020 16th International Conference on Network and Service Management (CNSM)*, pp. 1–6, IEEE, 2020.

[4] T. Goldschmidt, M. K. Murugaiah, C. Sonntag, B. Schlich, S. Biallas, and P. Weber, "Cloud-based control: A multi-tenant, horizontally scalable soft-PLC," *2015 IEEE 8th International Conference on Cloud Computing*, pp. 909–916, 2015.

[5] T. Hegazy and M. Hefeeda, "Industrial automation as a cloud service," *IEEE Trans. Par. and Distr. Syst.*, vol. 26, no. 10, pp. 2750–2763, 2015.

[6] B. Johansson, M. Rågberger, T. Nolte, and A. V. Papadopoulos, "Kubernetes orchestration of high availability distributed control systems," in *Proc. ICIT*, 2022.

[7] B. Satzger, A. Pietzowski, W. Trumler, and T. Ungerer, "A new adaptive accrual failure detector for dependable distributed systems," in *In ACM Symposium on Applied Computing (SAC 2007*, pp. 551–555, 2007.

[8] "Abb ability™ system 800xa control and i/o overview." https://library.e.abb.com/public/1a8bb0db5cf4474a95a113f96c18a5c2/3BSE047351%20en%20K%20ABB%20Ability%20System%20800xA%20Control%20and%20IO%20Overview.pdf. Accessed: 2023-03-20.

[9] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *Acm Sigact News*, vol. 33, no. 2, pp. 51–59, 2002.

[10] E. Brewer, "Cap twelve years later: How the" rules" have changed," *Computer*, vol. 45, no. 2, pp. 23–29, 2012.

[11] E. A. Lee, S. Bateni, S. Lin, M. Lohstroh, and C. Menard, "Trading off consistency and availability in tiered heterogeneous distributed systems," *Intelligent Computing*, vol. 2, p. 0013, 2023.

[12] E. A. Lee, R. Akella, S. Bateni, S. Lin, M. Lohstroh, and C. Menard, "Consistency vs. availability in distributed real-time systems," *arXiv preprint arXiv:2301.08906*, 2023.

[13] T. D. Chandra and S. Toueg, "Unreliable failure detectors for reliable distributed systems," *J. ACM*, vol. 43, pp. 225–267, Mar. 1996.

[14] Wei Chen, S. Toueg, and M. K. Aguilera, "On the quality of service of failure detectors," *IEEE Transactions on Computers*, vol. 51, pp. 13–32, Jan 2002.

[15] C. Fetzer, M. Raynal, and F. Tronel, "An adaptive failure detection protocol," in *Proceedings 2001 Pacific Rim International Symposium on Dependable Computing*, pp. 146–153, Dec 2001.

[16] M. G. Gouda and T. M. McGuire, "Accelerated heartbeat protocols," in *Proceedings. 18th International Conference on Distributed Computing Systems (Cat. No.98CB36183)*, pp. 202–209, May 1998.

[17] M. Van Steen, *Distributed systems*. Citeseer, 2017.

[18] A. Charny, R. Goguen, C. Iturralde, E. Hochberg, and J. P. Vasseur, "Distinguishing between link and node failure to facilitate fast reroute," July 26 2011. US Patent 7,986,618.

[19] C. Filsfils and D. D. Ward, "Technique for distinguishing between link and node failure using bidirectional forwarding detection (bfd)," Dec. 20 2011. US Patent 8,082,340.

[20] "Internet engineering task force (ietf) - bidirectional forwarding detection (bfd)." https://datatracker.ietf.org/doc/html/rfc5880#page-14. Accessed: 2022-06-03.

[21] H. Ritter, R. Winter, and J. Schiller, "A partition detection system for mobile ad-hoc networks," in *2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004.*, pp. 489–497, IEEE, 2004.

[22] H. Garcia-Molina, "Elections in a distributed computing system," *IEEE Trans. Comput.*, vol. 31, pp. 48–59, Jan. 1982.

[23] A. Arghavani, E. Ahmadi, and A. T. Haghighat, "Improved bully election algorithm in distributed systems," in *ICIMU 2011 : Proceedings of the 5th international Conference on Information Technology Multimedia*, pp. 1–6, Nov 2011.

[24] M. Khan, N. Agarwal, S. Jaiswal, and J. A. Khan, "An announcer based bully election leader algorithm in distributed environment," in *Smart and Innovative Trends in Next Generation Computing Technologies* (P. Bhattacharyya, H. G. Sastry, V. Marriboyina, and R. Sharma, eds.), (Singapore), pp. 664–674, Springer Singapore, 2018.

[25] S.-H. Lee and H. Choi, "The fast bully algorithm: For electing a coordinator process in distributed systems," in *Information Networking: Wireless Communications Technologies and Network Applications* (I. Chong, ed.), (Berlin, Heidelberg), pp. 609–622, Springer Berlin Heidelberg, 2002.

[26] L. Lamport, "The part-time parliament," in *Concurrency: the Works of Leslie Lamport*, pp. 277–317, 2019.

[27] L. Lamport, "Paxos made simple," *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001)*, pp. 51–58, 2001.

[28] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 USENIX Annual Technical Conference (Usenix ATC 14)*, pp. 305–319, 2014.

[29] R. E. Lyons and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability," *IBM journal of research and development*, vol. 6, no. 2, pp. 200–209, 1962.

[30] E. Dubrova, *Fault-tolerant design*. Springer, 2013.

[31] B. Johansson, M. Rågberger, A. V. Papadopoulos, and T. Nolte, "Heartbeat bully: Failure detection and redundancy role selection for network-centric controller," in *IECON*, 2020.