PAPER

Extended Dual Virtual Paths Algorithm Considering the Timing Requirements of IEC61850 Substation Message Types

Seokjoon HONG[†], Student Member, Ducsun LIM[†], Nonmember, and Inwhae JOE^{†a)}, Member

SUMMARY The high-availability seamless redundancy (HSR) protocol is a representative protocol that fulfills the reliability requirements of the IEC61850-based substation automation system (SAS). However, it has the drawback of creating unnecessary traffic in a network. To solve this problem, a dual virtual path (DVP) algorithm based on HSR was recently presented. Although this algorithm dramatically reduces network traffic, it does not consider the substation timing requirements of messages in an SAS. To reduce unnecessary network traffic in an HSR ring network, we introduced a novel packet transmission (NPT) algorithm in a previous work that considers IEC61850 message types. To further reduce unnecessary network traffic, we propose an extended dual virtual paths (EDVP) algorithm in this paper that considers the timing requirements of IEC61850 message types. We also include sending delay (SD), delay queue (DQ), and traffic flow latency (TFL) features in our proposal. The source node sends data frames without SDs on the primary paths, and it transmits the duplicate data frames with SDs on the secondary paths. Since the EDVP algorithm discards all of the delayed data frames in DQs when there is no link or node failure, unnecessary network traffic can be reduced. We demonstrate the principle of the EDVP algorithm and its performance in terms of network traffic compared to the standard HSR, NPT, and DVP algorithm using the OPNET network simulator. Throughout the simulation results, the EDVP algorithm shows better traffic performance than the other algorithms, while guaranteeing the timing requirements of IEC61850 message types. Most importantly, when the source node transmits heavy data traffic, the EDVP algorithm shows greater than 80% and 40% network traffic reduction compared to the HSR and DVP approaches, respectively.

key words: IEC61850, high-availability seamless redundancy, timing requirements, dual virtual paths, extended dual virtual paths, sending delay, delay queue, traffic flow latency, traffic reduction

1. Introduction

IEC61850 [1] is a communication protocol standardized by the International Electrotechnical Commission (IEC) for the design of an electrical substation automation system (SAS) in a smart grid. To ensure device interoperability, IEC61850 defines the complete communication architecture and data model from the process level to the station level [2].

IEC61850 divides the SAS into three communication levels: the process level, bay level, and station level, as shown in Fig. 1 [3]. The process level includes devices such as merging units, sensors (automatic reclosure, voltage control), and actuators (arc detection, measuring current, and voltage transformation). A merging unit converts the signals from the unconventional instrument transformers for the current and voltage or the analogue values from the con-

Fig. 1 IEC61850 communication architecture.

ventional instrument transformers into the IEC61850 messages. The bay level includes intelligent electronic devices (IEDs) that control and protect devices at the process level. At the station level, a station computer with a human machine interface (HMI) is used to control and monitor the system.

For the different roles of the devices at each level of the SAS, different types of messages are used, as shown in Fig. 1. The generic object oriented substation event (GOOSE) message exchanges information between the IEDs. This message is used for transmission of time-critical information such as interlocking and trips. The generic substation status event (GSSE) message is identical to GOOSE in terms of implementation; however, it transmits information about the state changes. The sampled value (SV) is used for transmission of the current and voltage samples. The manufacturing messaging specification (MMS) is used for client-server communications from the process level to the station level. The client can be an operating or monitoring system, and the server can be real devices or entire systems. Finally, the TimeSync message provides coordinated universal time (UTC) for synchronization. This time synchronization is based on the simple network time protocol (SNTP) version 4 [4]. The TimeSync message is transmitted to applications in the server and client substation IED.

In the IEC61850 standard, the message type and timing requirement for each IEC61850 message are defined as

Manuscript received October 6, 2015.

Manuscript revised February 2, 2016.

Manuscript publicized March 7, 2016.

[†]The authors are with the Hanyang University, Seoul, Korea.

a) E-mail: iwjoe@hanyang.ac.kr (Corresponding author)

DOI: 10.1587/transinf.2015EDP7410

Table 1 Types and timing requirements of IEC61850 messages.

Message	Type	Speed	Timing requirement
GOOSE/ GSSE	1A	Extremely fast	3–10 ms
	1B	Fast	20–100 ms
SV	4	Extremely fast	3–10 ms
MMS	2	Medium	≤100 ms
	3	Slow	≤500 ms
	5	Slow	≥1000 ms
TimeSync	6	Require accuracy	Not defined

Fig. 2 IEC61850 Communication Stack.

shown in Table 1 [5]. GOOSE and GSSE messages are implemented using Type1 (1A and 1B), SV messages are implemented using Type4, and MMS uses Type2, Type3 or Type5. Finally, the TimeSync message uses Type6.

Table 1 shows that the GOOSE/GSSE and SV messages require less latency than other messages.

Figure 2 illustrates the IEC61850 communication stack [6]. Since GOOSE and SV messages are time-critical, these applications are mapped directly to the Ethernet layer. MMS is an application for real devices and functions using TCP/IP layers and a client-server model. Finally, the TimeSync application uses a UDP/IP layer to broadcast TimeSync messages to all IEDs and corresponding devices.

To ensure a constant power supply in the transmission and distribution grid, the network for SAS has to fulfill stringent reliability requirements [7]. Therefore, SAS systems must be able to recover from a failure in less than 4–100 ms according to the IEC Technical Committee 57 (TC57) Working Group 10 (WG10). When the station bus transmits only the command information, delays up to 100 ms are tolerated. However, when time critical messages are carried, a delay of only up to 4 ms is tolerated.

Although the rapid spanning tree protocol (RSTP, IEEE 802.1D) is used for fast recovery in the Ethernet, it can only guarantee a recovery time of less than one second [8]. This makes the protocol unsuitable for the SAS. IEC61850 specifies two network redundancy protocols to meet the reliability requirements for the SAS. These two protocols are defined in the IEC62439-3 standard [9]. One is the parallel redundancy protocol (PRP), and the other is the high-availability seamless redundancy (HSR) protocol. Both of these protocols can overcome the failure of a link or node with zero recovery time. In PRP, the double attached node

Fig. 3 HSR principle and frame.

implementing PRP (DANP) uses the two networks simultaneously, and this provides for zero recovery time. In contrast, the HSR applies the PRP principle in a ring.

Figure 3 shows the HSR principle and HSR tag in the HSR frame. The fields in the HSR tag are as follows.

- HSR- ET (EtherType): This is used to identify the HSR frames.
- Path ID (4-bit): In an HSR ring, this indicates the direction (clockwise or counter clockwise). In general, it indicates the sending port.
- Size (12-bit): This is the size of the Link Service Data Unit (LSDU).
- Sequence Number (SN): This is used to detect duplicates and circulating frames. Each node increments the SN fields monotonically for each frame sent.

For each frame sent from the application, a source double attached node implementing HSR (DANH) must send two copies with an HSR tag over ports A and B. After DANH sends the duplicate frames, these frames circulate in opposite directions in the ring network, and every node forwards frames from one port to another, except if already forwarded. When a destination DANH receives a frame, it removes the HSR tag and passes the frame to the upper layer discarding any duplicates. Duplicates and circulating frames can be recognized by the doublet {source address, SN in HSR tag} at a receiver or forwarding node. Therefore, a receiver or a forwarding node must store the information about each received frame in memory in order to detect whether the node has received the frame. Through this HSR principle, HSR can provide seamless redundancy, even when there is a link failure in a ring network. Although HSR represents a suitable protocol in terms of reliability by providing seamless redundancy, it has the drawback of creating unnecessary redundancy traffic due to the duplicate frames.

There are a number of approaches that can be used to reduce redundant traffic in HSR networks. Shin and Joe proposed an algorithm that reduces the network traffic and retains the availability of the HSR network by using the Traffic Control IED [10]. Abdulsalam and Rhee proposed an approach called Port Locking (PL) to reduce extra traf-

fic in the HSR connected ring based network [11]. Nasaif and Rhee have proposed several approaches. They first presented two approaches called quick removing (QR) and virtual ring (VRing) [12]. The QR method is proposed for ring-type networks, while the VRing divides any closed-loop HSR into several VRings for traffic reduction. Secondly, they proposed an approach known as removing unnecessary redundant traffic (RURT) [13]. This removes these traffic type at an early stage instead of circulating it in the network. Recently, they presented the DVP algorithm to solve unnecessary HSR traffic problems [14]. The main concept of the DVP algorithm is to establish two virtual paths between a source DANH and a destination DANH. Because this algorithm uses these virtual paths rather than the standard HSR transmission process, it can reduce much of the unnecessary unicast traffic compared to the standard HSR.

Figure 4 shows a sample network consisting of four DANHs and four quadruple port devices (Quadboxes). DANH is used to send duplicate frame copies on separate links. The Quadbox is used to connect HSR based rings or networks. In HSR, if the QuadBox receives frames, it duplicates and forwards the frames randomly. In contrast, the DVP algorithm uses dual virtual paths to send unicast traffic by using the final paths (FP) table stored in the QuadBox. Although the DVP algorithm dramatically reduces unnecessary network traffic, this algorithm does not consider the timing requirements of the IEC61850 message type. Previously, we proposed a novel packet transmission (NPT) algorithm considering IEC61850 message types in order to reduce network traffic [15]. However, we only included the HSR ring network. Figure 5 shows the difference between

the HSR and NPT algorithm. In HSR, the source DANH sends duplicate frames with the same transmission period on both ports. In contrast, the source DANH in NPT sends duplicate frames with different periods on each port. On port A, which is near the destination DANH, the source DANH sends frames with a default transmission period as HSR, and on port B, the DANH sends frames with an extended transmission period according to the timing requirement of IEC61850 message types in order to reduce network traffic [15].

In this paper, we present an EDVP algorithm that considers the timing requirements of IEC61850 message types. In Sect. 2, we introduce an EDVP algorithm based on the DVP algorithm. In Sect. 3, we evaluate the performance of the EDVP algorithm. In Sect. 4, we discuss the traffic performance of the EDVP algorithm based on the results of all simulations. Finally, we present our conclusions in Sect. 5.

2. Extended Dual Virtual Paths Algorithm

In this section, we introduce an EDVP algorithm based on the DVP algorithm. As the DVP algorithm is only applicable to unicast traffic, the EDVP algorithm is designed for unicast traffic. The main idea of the EDVP algorithm is applying SD through DQ to the redundant data traffic flow in order to reduce the unnecessary network traffic. Here, the network traffic indicates the number of data frames in a network per unit of time, and the data traffic flow is used to designate the sequence of data frames that flow from the source node to the destination node. To apply SD to the specific data traffic flow, the EDVP algorithm needs to designate the primary and secondary paths among the dual virtual paths according to the priorities of the data traffic flows. The SDs for all the data traffic flows can be calculated using the timing requirements of the IEC61850 data traffic flow types and traffic flow latencies (TFLs). We defined the IEC61850 data traffic flow types as listed in Table 2. TFLs can be set using the current latencies of the data frames in the traffic flow according to the data traffic flow types. After calculating the SDs, the EDVP algorithm applies these SDs through the DQs to the data traffic flows on the secondary paths. Finally, the EDVP algorithm discards all the delayed data frames in the DQs when there is no link or node failure.

In Sect. 2.1, we describe how the EDVP algorithm designates the primary and secondary paths in dual virtual paths according to the priorities of all the data traffic flows. In Sect. 2.2, we explain how the EDVP algorithm applies the SDs for all data traffic flows on secondary paths according to the data traffic flow priorities and TFLs. We also describe

Fig. 4 HSR sample network.

Fig. 5 HSR ring network.

Table 2 Priorities and maximum latencies according to the IEC61850 data traffic types.

IEC61850 Data Traffic Flow Type(Tf(i))	SV, GOOSE/ GSSE (Tf(0))	MMS(Type2) (Tf(1))	MMS(Type3) (Tf(2))
Priority	High	Medium	Low
Max latency(L_{max})	10 ms	100 ms	500 ms

Fig. 6 Difference in transmission of the data frames from the source node between the existing algorithm and EDVP algorithm.

how the EDVP algorithm discards redundant data frames using DQs in order to reduce network traffic.

2.1 Primary and Secondary Paths According to the Priorities of IEC61850 Data Traffic Flow Types

First, two virtual paths are established as in the DVP algorithm. The EDVP algorithm then designates the primary and secondary paths in the dual virtual paths according to the data traffic flow priorities. We define the priorities and maximum latencies of the IEC61850 data traffic flow types, as provided in Table 2 based on the IEC61850 message types and timing requirements as defined in Table 1.

If the priority of the data traffic flow is high, the fastest path of the dual virtual paths (DVP) is designated as the primary path, and the second path of the DVP is designated as the secondary path for this data traffic flow. If the priority of the data traffic flow is medium or low, the second path of the DVP is designated as the primary path, and the fastest path of the DVP is designated as the secondary path for this data traffic flow. For example, if DANH1 sends data traffic to DANH2 as shown in Fig. 4(b), then Path1 is set as the primary path and Path2 is set as the secondary path for high priority data traffic flow such as SV and GOOSE/GSSE messages, and Path2 is set as the primary path and Path1 is set as the secondary path for medium or low priority data traffic flow such as the MMS (Type2, 3, and 5) messages.

2.2 SDs for All the Data Traffic Flows, According to IEC61850 Data Traffic Flow Types and TFL

After the EDVP algorithm designates the primary and secondary paths according to data traffic flow priorities, it applies SDs through the DQs to all of the data traffic flows on the secondary path in order to reduce the network traffic.

Figure 6(b) shows the concept of the SD in the EDVP algorithm. In this figure, we introduce a special queue known as the delay queue (DQ) to delay sending the frames of the same data traffic flow type from the source node.

For periodic incoming packets, in existing algorithms such as the HSR and DVP algorithm, the source node sends the data frames directly to the transmit queue for distribution to the network. In contrast, in the EDVP algorithm, the source node first sends data frames to the DQ. In the DQ, the

data frames are delayed until a specified time known as SD has elapsed. Then, the data frames are sent to the network from the DQ.

In the existing algorithm, the transmission delay (D_{Trans}) and the data rate (R_{D_1}) can be calculated using Eqs. (1) and (2), respectively, where L is the size of the frame in bits and FI is the frame interval.

$$D_{Trans}(sec) = \frac{L(bits)}{Link\ Bandwidth(bps)} \quad (1)$$

$$R_{D_1}(bps) = \frac{L(bits)}{FI(sec)} \quad (2)$$

Therefore, if the sizes of the data frame and link bandwidth are the same in both algorithms (the existing and the EDVP algorithm), the transmission delay in these models will also be the same. In the EDVP algorithm, the data rate (R_{D_2}) can be calculated using Eq. (3).

$$R_{D_2}(bps) = \frac{L(bits)}{FI + SD(sec)} \quad (3)$$

For example, if the SD is the same as the FI, then the data rate after applying SD will be half the data rate of the existing algorithms. Therefore, the data rate can be reduced by applying SD.

In addition, if N_{TrQ_1} and N_{TrQ_2} indicate the total number of frames that can be transmitted through the transmit queue per second by using the existing algorithm and the EDVP algorithm, respectively, then N_{TrQ_1} and N_{TrQ_2} can be expressed as follows.

$$N_{TrQ_1} = \frac{1}{FI} \quad (4)$$

$$N_{TrQ_2} = \frac{1}{FI + SD} \quad (5)$$

Therefore, in the EDVP algorithm, the total number of frames being delayed in the DQ for one second (N_{DQ}) can be calculated using Eq. (6).

$$N_{DQ} = N_{TrQ_1} - N_{TrQ_2} \quad (6)$$

For example, if both FI and SD are 0.1, then N_{TrQ_1} is 10, N_{TrQ_2} is 5, and N_{DQ} is 5.

The EDVP algorithm does not apply the SD to the primary path; thus, the traffic on the primary path is transmitted equally, as in the DVP algorithm. The SD for a specific data traffic flow is calculated at the source node according to the IEC61850 data traffic flow type and TFL. Even though the SD is applied only to the redundant data traffic flow on the secondary path, the timing requirements of the IEC61850 messages must be considered for safe message transmission, even when a link or node failure occurs in the primary path. Because of this, the SD for the data traffic flow must be less than the maximum latency (L_{max}) of the data traffic flow ($Tf(i)$). In the EDVP algorithm, we consider IEC61850 data traffic flow types and maximum latencies to calculate the SD, as shown in Table 2.

The TFL of the specific data traffic flow is calculated


```

/* Initialization of L_curr (Tf(i)) variable */
L_curr (Tf(i)) = 0

/* Initialization state variables */
READY_TO_SEND_TFLC = FALSE /* To check whether it is ready to send TFLC frame or not */
TRAFFIC_FLOW = FALSE /* To check whether there is data traffic flow or not */
TFLC_PERIOD_EXPIRED = FALSE /* To check whether TFLC frame period is expired or not */

IF ( TRAFFIC_FLOW == TRUE ) THEN
    TEMP_LATENCY(Tf(i)) = CURRENT_TIME - SOURCE_TIME_STAMP(Tf(i))
ENDIF

WHILE (TFLC_PERIOD_EXPIRED == FALSE) DO
    IF (TEMP_LATENCY(Tf(i)) > L_curr (Tf(i))) THEN
        L_curr (Tf(i)) = TEMP_LATENCY(Tf(i))
    ENDIF
ENDWHILE

READY_TO_SEND_TFLC = TRUE
(a) L_curr calculation of data traffic flow type i (Tf(i))

/* Initialization of TFL (Tf(i)) variable */
TFL (Tf(i)) = 0

IF ( READY_TO_SEND_TFLC == TRUE) THEN
    IF (L_curr (Tf(i)) <  $\alpha$  * L_max (Tf(i))) THEN
        TFL(Tf(i)) = 1 /* Value 1 indicates that traffic flow latency is 'Low' */
    ELSE IF (  $\alpha$  * L_max(Tf(i))  $\leq$  L_curr (Tf(i)) ) AND (L_curr (Tf(i)) < L_max (Tf(i))) THEN
        TFL(Tf(i)) = 2 /* Value 2 indicates that traffic flow latency is 'Stable' */
    ELSE IF (L_max (Tf(i)) < L_curr (Tf(i)) ) THEN
        TFL(Tf(i)) = 3 /* Value 3 indicates that traffic flow latency is 'Alert' */
    ELSE
        TFL (Tf(i)) = 0 /* Value 0 is used for other case not to apply SD. */
    ENDIF

    Set TFL (Tf(i)) value in the TFLC frame
    Send TFLC frame to the destination through the primary path
    READY_TO_SEND_TFLC = FALSE
    L_curr (Tf(i)) = 0
ENDIF
(b) TFL(Tf(i)) setting according to L_curr (Tf(i))

```

Fig. 7 Pseudo code for L_{curr} calculation of data traffic flow type i and TFL setting in the TFLC frame at the destination

at the destination node. The value of the TFL is set according to the current latency (L_{curr}) of the data traffic flow ($Tf(i)$). The pseudo codes in Fig. 7 (a) and (b) describe the $L_{curr}(Tf(i))$ calculation and TFL($Tf(i)$) setting at the destination node, respectively.

where α is a tunable value ($0 < \alpha < 1$) that can be configured by the system manager.

The current latency of the data frames in the flow can be measured whenever a destination node receives data frames from the source node. In general, all the nodes in the IEC61850-based SAS network can be synchronized using a simple network time protocol (SNTP), as shown in Fig. 1. Therefore, the latency of the data frames in the flow can be measured using the time stamp of the data frames [16]. In addition, if the IEEE 1588 standard [17] is applied in the SAS, high-precision time synchronization can be used to more accurately measure the latency.

The $L_{curr}(Tf(i))$ value can be set as the maximum value among the recent latency values to apply the SD and guarantee the timing requirement of each message at the source node. The TFL($Tf(i)$) value can be either “1 (Low)”, “2 (Stable)”, or “3 (Alert)”, according to the condition of $L_{curr}(Tf(i))$. The meaning of each TFL value is as follows: “1 (Low)” indicates that the source node can further extend the SD, “2 (Stable)” indicates that the source node must fix SD as the current value, and “3 (Alert)” indicates that the source node must reduce the SD because the current latency is already greater than the maximum latency of the data traffic flow type. To carry this value from the destination node

Fig. 8 Traffic flow latency in the TFLC frame

Table 3 The bit setting and meaning according to the TFL value

TFL Value	Bit	Meaning
0	00	None
1	01	Low
2	10	Stable
3	11	Alert

to the source node, we propose a traffic flow latency checking (TFLC) frame.

The TFLC frame format is defined as shown in Fig. 8. We suggest that the frame size of the TFLC be fixed at 64-bytes (minimum Ethernet frame size) to minimize the use of network bandwidth. The values of the TFL for different traffic flow types can be encoded in a latency field in the TFLC frame, as shown in Table 3.

Using the TFLC frame, the destination node can provide information about the TFL for all data traffic flow types, and the TFLC frames are sent through the primary path of each data traffic flow. We can also use this frame to detect the link or node failures in the dual virtual paths. To immediately detect a failure using TFLC frames, we suggest a period of 10 ms for the TFLC frame. If the source node does not receive the next TFLC frame during this period, the source node immediately stops applying the SD on the secondary path in order to transmit data frames safely, guaranteeing the timing requirements of the IEC61850 message types.

When the source node receives a TFLC frame, it checks the TFL value in the frame and calculates the SD according to the TFL and L_{EXP} . Here, L_{EXP} is the expected latency when the source node sends data frames to the destination node. In general, the end-to-end delay (D_{ETE}) between two nodes is defined as in Eq. (7). D_{Trans} represents the transmission delay; D_{Prop} is the propagation delay; and D_{Proc} indicates the processing delay. L_{EXP} can then be calculated, as in Eq. (8).

$$D_{ETE} = D_{Trans} + D_{Prop} + D_{Proc} \quad (7)$$

$$L_{EXP} = D_{ETE} + SD \quad (8)$$

Initially, the SD value for all of the data traffic flows is set to zero because the destination node has not yet received any data frames from the source node. After sending data frames to the destination node, the source node can receive

```

/*Initialization of SD (Tf(i)) variable */
SD(Tf(i)) = 0
/*Initialization of state variable for checking whether it is first time that node receives alert or not*/
FIRST_TIME_ALERT = TRUE

IF (  $L_{EXP}(Tf(i)) < \alpha * L_{max}(Tf(i))$  AND (TFL(Tf(i)) == 1) THEN
    SD(Tf(i)) = SD(Tf(i)) +  $\beta * L_{max}(Tf(i))$ 
ELSE IF (TFL(Tf(i)) == 2) THEN
    No operation
ELSE IF (TFL(Tf(i)) == 3) AND (FIRST_TIME_ALERT == TRUE) THEN
    SD(Tf(i)) = SD(Tf(i)) / 2
    FIRST_TIME_ALERT = FALSE
ELSE IF (TFL(Tf(i)) == 3) AND (FIRST_TIME_ALERT == FALSE) THEN
    SD(Tf(i)) = 0
    FIRST_TIME_ALERT = TRUE
ELSE
    SD(Tf(i)) = 0 /* For other case, source node does not apply sending delay. */
ENDIF

```

Fig. 9 Pseudo code for SD calculation at the source node

Fig. 10 Process of sending IEC61850 frames at the source node

TFLC frames from the destination node. As shown in Fig. 9, if $L_{EXP}(Tf(i))$ is less than $\alpha * L_{max}(Tf(i))$ and the TFL(Tf(i)) value in TFLC is “1 (Low),” the source node can extend the SD for each data traffic flow using the formula with α and β . Here, the α value is the same value used to set the TFL at the destination node. The tunable value β sets the degree of increase at a time. To do this, it must be positive and less than α ($0 < \beta < \alpha$). We suggest $\alpha=0.9$ and $\beta=0.1$ as the default values. If the destination sends “2 (Stable)” through the TFLC frame, the source node stops increasing the SD and keeps sending traffic with the current SD. The source node can also receive a “3 (Alert)” from the destination node because of network congestion or other problems. In that case, the source node should reduce the SD. Therefore, we propose that the SD must be halved when the source node receives a “3 (Alert);” if it receives another “3 (Alert),” SD must be set to zero in order to send the data frames safely.

Figure 10 shows how the source node sends data traffic on the primary and secondary paths. If there are incoming frames in Layer 2 from the application on the source node, first the EDVP duplicates the frames and attaches the HSR tag. Here, we suggest that “A”-frames are sent on the primary path and “B”-frames are transmitted on the secondary path.

On the secondary path, the source node uses the DQ to delay sending the data frames according to the data traffic flow priorities. Once EDVP classifies the data frames, it enqueues them to the different DQs based on the data traffic flow priorities. EDVP then checks the TFL in the TFLC

frame and calculates the SD. Later, EDVP applies the SD and enqueues the data frames to the transmit queue in order to send data frames to the destination. When the destination node receives the duplicate frames through the dual virtual paths from the source node, it removes the HSR tag and passes the frames to the upper layer discarding any duplicates as in the HSR.

By applying the SD, the source node can reduce the data rate of the data traffic flow on the secondary path. In addition, whenever the source node receives TFLC through the primary path, it discards all the delayed data frames (“B”-frames) in the DQ because the destination node receives duplicate data frames (“A”-frames) through the primary path when there is no link or node failure. Thus, the reduced number of redundant frames (N_{rf}) during one second using EDVP, compared with the DVP algorithm, is equal to the sum of the number of delayed frames during one second in the DQs at all source nodes in the network. Since there are three DQs according to the traffic flow types at the source node, N_{rf} can be calculated as in Eq. (9).

$$N_{rf} = \sum_{i=1}^m \sum_{j=0}^2 N_{DQ}(Tf(j)) \quad (9)$$

Here, i is the data traffic flow, m is the total number of data traffic flows in the network, and $N_{DQ}(Tf(j))$ is the number of frames being stored in the DQ for traffic flow type j for one second. Therefore, if the total number of frames being transmitted per second in the network by using the DVP and EDVP algorithms are N_{DVP} and N_{EDVP} , respectively, then N_{EDVP} can be calculated as in Eq. (10).

$$N_{EDVP} = N_{DVP} - N_{rf} \quad (10)$$

In this way, not only is the data rate of the data traffic flow reduced, but the network traffic can be reduced by using the EDVP algorithm.

If a link failure occurs in the network under the EDVP algorithm, this indicates that there are failures both on the primary path and the secondary path for different data traffic flow types. However, EDVP can adjust to this and provide reliability as follows.

1. Failure on the secondary path: The destination node can receive all duplicate frames through the primary path as in HSR.
2. Failure on the primary path: The source node immediately sends all the delayed frames in the DQ through the secondary path and starts sending current data frames without the SD. Then the destination node can receive all the duplicate frames.

Figure 11 details the total flow of the EDVP algorithm at the source node and the destination node. Figure 12 shows the difference in data traffic flow on dual virtual paths between the DVP and the EDVP algorithms. Finally, Table 4 lists the primary and secondary paths under the EDVP algorithm illustrated in Fig. 12 (b).

Fig. 11 Total flow of the EDVP algorithm

Fig. 12 Network traffic difference between DVP and EDVP

Table 4 The primary and secondary path according to the data traffic flow priority

Data traffic flow priority	High priority traffic flow	Med/Low priority traffic flow
Path1 (D1-Q2-D2)	Primary path	Secondary path
Path2 (D1-Q1-Q4-Q3-D2)	Secondary path	Primary path

3. Performance Evaluation

In this section, we show the performance of the EDVP algorithm using the OPNET network simulator [18] in terms of network traffic. We designed two different network models to evaluate our algorithm and to compare it with the standard HSR, NPT, and DVP algorithms. We implemented all of the

functions of the DANH and QuadBox in the network model using C programming for the four algorithms: the standard HSR, the NPT algorithm, the DVP algorithm, and the EDVP algorithm.

The implemented functions of DANH and QuadBox are as follows. In the HSR algorithm, DANH [9] has two HSR ports and sends the duplicated frames on separate links when it receives frames from applications. QuadBox [9] has four HSR ports and is used to interconnect two rings or networks. If QuadBox receives frames from one port, it duplicates and forwards frames to the other ports. However, if it receives the same frame from the same port, it discards that frame. In the NPT algorithm, the source DANH sends duplicate frames with different periods on each port as shown in Fig. 5 of Sect. 1. The QuadBox under the NPT is the same as in the HSR algorithm. In the DVP algorithm [14], the function of DANH is the same as in the HSR algorithm, whereas the QuadBox under the DVP algorithm has information on each virtual path for each connection pair in the final path (FP), and it forwards frames based on this table. Thus, each DANH node can send frames through dual virtual paths. Finally, the function of the QuadBox is the same in the EDVP algorithm as in the DVP algorithm. On the other hand, EDVP applies SD to the data traffic flows on the secondary paths at the source DANH according to the data traffic flow priorities.

The first network model is the same as the HSR sample network, which is shown in Fig. 4. In the first network model, we show how the SD is applied using the EDVP algorithm and how the SD is changed when a link failure occurs. In this network model, we also show how DQ can be used to reduce the network traffic in the EDVP algorithm. The second network model is modeled similar to a real IEC61850 based network. This network model is devised to evaluate the performance of the algorithm with respect to the number of DANHs in the bay rings and the source data traffic.

Since reducing network traffic is important in SAS network, we evaluate the performance of algorithms in terms of network traffic as in previous works [10]–[15]. For example, if algorithm A reduces more network traffic than algorithm B, it indicates that the traffic performance of algorithm A is better than that of algorithm B.

Table 5 shows the common simulation parameters for all of the network models. We assume that there is no propagation delay in the network. We also assume that all network devices are synchronized using the simple network time protocol (SNTP) or IEEE 1588 standard. We set a processing delay, as shown in Table 5, based on previous research, which showed that the processing time of the GOOSE message using a security algorithm on one side (publisher or subscriber) is about $700 \mu s$ [19]. Finally, we assume that all data traffic in the network is the unicast traffic.

3.1 The First Network Model

Figure 13 shows the first network model, and Table 6 shows

Table 5 Simulation parameters for all network models

Simulation Parameters	Values	
Link model	Full-duplex Ethernet	
Link bandwidth	100 Mbps	
Data rate of source traffic	1 Mbps (default)	
Data frame size	500 byte	
TFLC frame period (EDVP algorithm)	10 ms	
TFLC frame size	64 byte	
Processing delay	DANH	700 μ s
	QuadBox	40 μ s
Data traffic flow probability according to priorities (Based on uniform distribution)	High priority	0.4
	Medium priority	0.3
	Low priority	0.3

Fig. 13 The first network model**Table 6** Scenario for the first network models

Scenario	Connected pairs (src, dest)	Event	Simulation time
Failure-Free	(DANH1, DANH2)	No link Failure	0–60 s
	(DANH2, DANH4)		
	(DANH3, DANH1)		
	(DANH4, DANH3)		
Link-Failure	(DANH1, DANH2)	No link Failure	0–10 s
		Link Failure between DANH1 and QuadBox1	10–20 s
		No link Failure	20–30 s
		Link Failure between QuadBox3 and QuadBox4	30–40 s
		No link Failure	40–60 s

the simulation scenarios.

In the Failure-Free scenario, we simulated the network model with four connection pairs using the NPT, DVP and EDVP algorithm. The three results in Fig. 14 show the end-to-end delay of each priority traffic flow, which is measured at the destination node when DANH1 sends the data traffic to DANH2 in the network model. For different data traffic flows under the NPT(“B”-frame) and EDVP algorithm with the constant values α and β , the end-to-end delays change for a short time, and then all the delays have a constant value for the entire range of the simulation time.

In this network model, DVP and EDVP use the same dual virtual paths to send data traffic from DANH1 to DANH2. The fastest path of the dual virtual paths (DVP) is Path1, and the second path of DVP is Path2.

Fig. 14 End-to-end delay according to traffic flow priority

The route of Path1 is DANH1-QuadBox1-DANH2, and the route of Path2 is DANH1-QuadBox2-QuadBox4-QuadBox3-DANH2. In the DVP algorithm, the end-to-end delays on Path1 and Path2 seem to be almost the same, although Path2 has more intermediate nodes than Path1. This is because we set the processing delay at the intermediate node (QuadBox) to be much less than the processing delay at the end node (DANH), as shown in Table 5. In this algorithm, the data frame can arrive at the destination using both paths in less than 2 ms according to Eq. (7). This is because the total processing delay is about 1.5 ms and the transmission delay is only 40 μ s, according to Eq. (1). In contrast, in the EDVP algorithm, the end-to-end delays for the data traffic flows on the secondary path are increased until they reach about $\alpha * L_{max}(Tf(i))$, as shown in Table 2. Here, α and β are set as shown in the graphs in Fig. 14. As explained in Sect. 2, whenever the destination node receives the data frames from the source node, it calculates the current latency(L_{curr}) of the data traffic flow ($Tf(i)$) as shown in Fig. 7 (a). The destination node then sets the TFL values for each data traffic flow as shown in Fig. 7 (b). Finally, the destination node carries the TFL values of the source node through the TFLC frame every 10 ms. In addition, $SD(Tf(i))$ is extended as shown in Fig. 9. In this network simulation, we assume that there is no network congestion, so that the source node maintains an SD value for specific data traffic when it receives the TFLC frame, and the TFL for the specific data traffic is “2 (stable)”. The graphs in Fig. 14 show that the secondary path for the high priority traffic flow is Path2, and the secondary path for the medium and low pri-

Fig. 15 Average network traffic in the first network model

ority traffic flow is Path1. We can also see that the SD can be changed using α and β in the EDVP algorithm, and that this affects the end-to-end delay. As we show through the graphs, α can be used to change the maximum SD, and β can be used to change the number of steps until the current SD reaches the maximum transmission delay.

Figure 14 also shows the end-to-end delay of each priority traffic flow in the NPT algorithm. In this simulation, we measured the end-to-end delay of only the first arrived frames (“A”-frames or “B”-frames) because many frames are duplicated by the Quadboxes. Since the source DANH under the NPT algorithm sends “A”-frame with a default transmission period to port A, which is near the destination as shown in Fig. 13 (a), the end-to-end delays of the “A”-frame are the same as those in DVP (Path1). On the other hand, the end-to-end delays of “B”-frame are extended according to the IEC61850 message types. In the NPT algorithm, whenever the source node sends frames, the transmission period is extended until it reaches $0.9 * L_{max}(Tf(i))$ [15]. However, we can see that the end-to-end delays of the high priority traffic flow exceeds $L_{max}(Tf(i))$ in Fig. 14 (a). This is because the NPT algorithm only considers delay at the source DANH, when it expands transmission period. Therefore, if other delay components such as the processing delay and transmission delay are added, it is possible that the end-to-end delay of the “B”-frame cannot meet the timing requirements of the IEC61850 message types.

In the Failure-Free scenario, we also simulated the network model to evaluate the EDVP algorithm in terms of network traffic compared to standard HSR, NPT, and DVP algorithms. In this simulation, the data traffic from the source node was increased by 1 Mbps every second. We calculated the network traffic in the network using the following equation.

$$Network\ traffic = \sum_{i=1}^n \sum_{j=1}^2 LT(i, j) \quad (11)$$

Here, $LT(i, j)$ is the j direction traffic at link i . The total number of links in the network is n . For example, if node A and node B are linked, $j = 1$ is the traffic from A to B, and $j = 2$ is the traffic from B to A. We can see that n is 12 in the first network model. In this paper, we measured the network traffic in bits per second (bps).

Figure 15 shows the average network traffic during the simulation under the standard HSR, NPT, DVP, and EDVP

Fig. 16 Accumulated reduced number of redundant frames using EDVP algorithm in comparison with DVP algorithm

Fig. 17 Average data and TFLC traffic in the first network model under EDVP algorithm.

algorithms. The average network traffic of the EDVP algorithm also includes the TFLC traffic. The results show that the traffic performance of the NPT algorithm is slightly better than that of the HSR algorithm. The reason is that the source DANH in the NPT algorithm sends data frames with an extended transmission period to a port that is far from the destination, whereas the source DANH in HSR sends duplicate frames with the default transmission period on both ports. Then, we can see that the traffic performances of the DVP and EDVP algorithms are much better than those of the standard HSR and NPT algorithm. This is because the data traffic is sent through the dual virtual paths under the DVP and EDVP algorithm, whereas many frames are duplicated and forwarded by the QuadBox under the HSR and NPT algorithms. The results also show that the EDVP algorithm more highly reduces network traffic than the DVP algorithm. The EDVP algorithm nearly halves the network traffic of the DVP.

Figure 16 shows the accumulated reduced number of redundant frames in the DQs at four source DANHs by using the EDVP algorithm. Here, the reduced number of redundant frames was calculated using Eq. (9). As we can see from this result, with larger α and β , the reduced number of redundant frames in the DQs was greater. Thus, we can understand why the network traffic using the EDVP algorithm with larger α and β was less than the DVP algorithm.

Figure 17 shows the average data traffic and TFLC traffic separately under the EDVP algorithm during the simulation. We can see that average TFLC traffic stayed constant during the simulation. This is because TFLC traffic does not depend on α , β , and source data traffic. The TFLC traffic depends on its frame size and period, which are set as in Table 5. The average TFLC traffic is about 1.2Mbps and this

Fig. 18 RP in the first network model.

is much less than the data traffic as shown in Fig. 17.

Figures 18 (a), (b), and (c) show the network traffic reduction percentage (RP) of the NPT compared to the HSR, and the RP of DVP and EDVP compared to HSR and DVP, respectively. Equation (12) is used to calculate the RP, where T_A and T_B are the network traffic of the algorithm A and the algorithm B, respectively. This equation was also used to evaluate the traffic performance in previous work [12].

$$RP \text{ of } A \text{ compared to } B = \left[1 - \frac{T_A}{T_B} \right] * 100\% \quad (12)$$

As shown in Fig. 18 (a), the RP of the NPT compared to HSR starts at about 20% and after 15s, it decreases gradually to about 10%. In Fig. 18 (b), the RP of DVP compared to HSR starts at about 75% and decreases continuously after 20 sec. In contrast, with EDVP, when α is 0.9, the RP compared to HSR is greater than 85% for 10–30 sec, and when α is 0.2, the RP is between 80% and 85% for 0–30 sec. We can see that the RP under all algorithms begins to decrease after 20 sec. This is because the amount of source data traffic (the data traffic sent from the source node) nearly reaches the link bandwidth after 20 sec. In Fig. 18 (c), the RP of EDVP with different α and β compared to DVP during the simulation is between 10% and 50%. The RP of EDVP with large α (0.9) compared to DVP became greater than 40% as the source data traffic increased.

Moreover, in the EDVP algorithm, by using a larger value for α , the network traffic can be more highly reduced. In contrast, changing β has only a slight effect. This means

Fig. 19 End-to-end delay of traffic on Path2 when a link failure occurs in Path1 (at 10 sec)

that if we use a larger value for α , in the range $(0 < \alpha < 1)$, a larger SD can be applied so that a greater number of frames can be delayed in the DQ. Since all the delayed frames in the DQs are discarded when there is no link or node failure, the network traffic can be more highly reduced by using a larger value for α . In addition, the β value can be used to change the amount of increase over one step until SD reaches the maximum value.

To determine how the EDVP algorithm operates when a link failure occurs, we simulated the network with one connection pair (DANH1 - DANH2) as shown in Fig. 13 (b). Two link failures occurred at two different links during the simulation, as shown in Table 6.

Figures 19 and 20 show the end-to-end delay according to the traffic flow priorities when a Path1 or Path2 link failure occurs, respectively, while DANH1 sends 1 Mbps or 10 Mbps of data traffic to DANH2. In order to show the changes in detail when failure occurs, the simulation results are shown for 5 seconds, 3 seconds before and 2 seconds after the failure on each path occurred. In Fig. 19, we can see a change in the end-to-end delay of the high priority traffic flow on Path2, and Fig. 20 shows a change in the end-to-end delay of the medium and low priority traffic flow on Path1. The source data traffic ratio according to traffic flow priorities is set as shown in Table 5, and we set α and β as 0.9 and 0.1 respectively. These results show that all the SDs for the source data traffic changed to zero after link failure. This indicates that the source node immediately stopped applying the SD to the data traffic flow on the secondary path because the source node could not receive the TFLC frame after link failure occurred. In Figs. 19 (b) and 20 (a) and (b), we can also see the transmission of data frames that look like a vertical line. This is because these data frames were sent immediately from the DQs when the link failure occurred.

Fig. 20 End-to-end delay of traffic on Path1 when a link failure occurs in Path2 (at 30 sec)

Fig. 21 The second network model (Number of DANH: 16).

In addition, because the source node needs to enqueue more data frames to the DQs when the source node sends 10 Mbps of data traffic, there were more frames sent from the DQs than when the source node sent 1 Mbps of data traffic.

All of the results collectively show that the EDVP algorithm can fulfill the timing requirements according to the traffic flow types, even when link failure occurs in the dual virtual paths.

3.2 The Second Network Model

Figure 21 shows the second network model when there are 16 DANHs (4 DANHs in each bay ring). To model the real IEC61850 based SAS network, the network consists of one station ring network and four bay ring networks. We also designed every ring network to link with

Fig. 22 Average network traffic in the second network model

Fig. 23 Network traffic reduction in the second network model

the other ring networks through two Quad Boxes, as specified in IEC62439-3 [9]. There are four source-destination pairs (DANH1- DANH13, DANH16-DANH12, DANH9-DANH8, and DANH5-DANH4) in the network. The parameters of data traffic flow are the same as in the first network model. We simulated the second network model with increasing numbers of DANHs in the bay ring.

Figure 22 shows the average network traffic according to the total number of DANHs in the bay rings under the standard HSR, NPT, and DVP, and EDVP algorithm ($\alpha=0.9$, $\beta=0.1$). The average network traffic of the EDVP algorithm includes the TFLC traffic. Figure 23 shows the traffic reduction in the second network model among the four algorithms. In Figs. 22 (a) and (b), we can see that network traffic

under all of the algorithms increased linearly as the number of DANH increased. In particular, the network traffic under the HSR and NPT algorithms increased more rapidly than under the other algorithms. As in the first network model, the NPT shows better traffic performance than the HSR. In Figs. 23 (a) and (b), the RP of NPT compared to HSR gradually increased until it reaches 10%. After that, it increased slightly as the number of DANH increased. When the source node sends 10Mbps data traffic, the RP of NPT increased to about 20%. However, DVP and EDVP show much better traffic performance than NPT algorithm. The DVP shows almost the same traffic performance compared to the standard HSR algorithm regardless of the source data traffic. In contrast, the EDVP shows better traffic performance when the source data traffic is increased.

The EDVP showed slightly better traffic performance than the DVP when the source sent 1 Mbps data traffic, as shown in Fig. 23 (a). We can see that the RP of DVP and EDVP compared to the HSR algorithm remained steady at greater than 70% and 75%, respectively, regardless of the number of DANH. However, when the source node sent 10 Mbps of data traffic, the traffic performance of the EDVP was better than that of the DVP, as shown in Fig. 23 (b). The traffic reduction was greater than 80% and 40% compared to HSR and DVP, respectively.

4. Discussion

The simulation results of the two network models show that the EDVP algorithm demonstrates better performance than the HSR, NPT, and DVP algorithms. We also confirmed that the DVP can achieve approximate a 70% reduction in traffic compared to the HSR algorithm, in agreement with a previous study [14].

Even though the DVP algorithm demonstrates much better performance than the HSR algorithm using dual virtual paths, it does not consider the timing requirements of the IEC61850 message types. However, the EDVP algorithm based on the DVP algorithm considers the timing requirements of the IEC61850 message types to further reduce unnecessary network traffic. In addition, although the NPT algorithm considers the IEC61850 message types, we can see that the end-to-end delay of “B”-frame in the first network model does not meet the timing requirements of the IEC61850 message types because the NPT does not consider processing and transmission delay. This indicates that the NPT does not guarantee the reliability for SAS network, when a link failure occurs.

In the first network model under the EDVP algorithm, we showed how SD can be applied according to traffic flow priority with different α and β values and how DQ can be used to reduce unnecessary network traffic. Because the larger value of α caused a larger SD, the EDVP algorithm with larger α showed better traffic performance. The β value can be used to adjust the amount of increase in one step until SD reaches the maximum values. Therefore, the α and β values for the specific data traffic flow can be set by the

system manager according to system requirements. In order to send current latency of each data traffic type and check link failure, TFLC traffic is additionally used in EDVP algorithm. However the TFLC frame is much less than the data traffic. The simulation results show that EDVP can reduce network traffic more than the other algorithms, even though it uses TFLC traffic. Furthermore, in this network model, the EDVP algorithm can fulfill the timing requirements of the IEC61850 message types, even when a link failure occurs. This is because the delayed data frames in the DQs can be sent immediately when a link failure occurs.

In the second network model simulation, we modeled a real IEC61850 based SAS network that includes one station ring network and four bay ring networks. In this network model, EDVP showed better traffic performance than the other algorithms. EDVP even showed greater than 80% and 40% network traffic reduction compared to the HSR and DVP algorithms, respectively, when the source data traffic was increased. In a real IEC61850 based SAS network, the DANHs of the second network model can be IEDs as shown in Fig. 1 of Sect. 1. Therefore, if there is heavy data traffic such as MMS and GOOSE between IEDs, better traffic performance can be achieved by using EDVP algorithm compared to those of the standard HSR and DVP algorithm.

5. Conclusion

In our paper, we proposed an EDVP algorithm considering the timing requirement of IEC61850 message types. The EDVP algorithm reduces unnecessary network traffic more than the DVP algorithm by applying SDs through DQs on secondary paths according to the IEC61850 traffic flow type.

We evaluated the EDVP algorithm using the OPNET network simulation with two types of network models. In the first network model, we showed how the EDVP algorithm applies SDs according to the IEC61850 traffic type and how the algorithm uses DQs in order to reduce unnecessary network traffic. The second network model is modeled similarly to a real IEC61850 based SAS network. Together, the simulation results show that the EDVP algorithm has better traffic performance than the standard HSR, NPT, and DVP algorithms. Therefore, if the EDVP algorithm is applied to a real IEC61850 network, better traffic performance can be achieved than is possible with the standard HSR, NPT, and DVP algorithms, while guaranteeing the timing requirements of IEC61850 traffic types.

Although the EDVP algorithm can reduce much more unnecessary network traffic than previous algorithms, the EDVP algorithm can only reduce the unicast traffic. This is because the EDVP algorithm is designed based on the DVP algorithm, which is only applicable to unicast traffic. Hence, improving the EDVP algorithm to reduce multi-broadcast traffic types would be worthwhile future work.

References

- [1] IEC 61850, “Communication networks and systems in substations,”

2nd ed.

- [2] H.-E. Olovson, T. Werner, and P. Rietman, "Next Generation Substations, Impact of the Process Bus," ABB Review, Special Report IEC61850, 2010.
- [3] R. Faria, A wireless sensor network for electronic distribution substation, Master's thesis, Universidade de Lisboa, 2011.
- [4] Internet RFC 2030 simple network time protocol (SNTP) Version 4 (1996) from IPv4, IPv6 and OSI.
- [5] IEC61850-5, Communication Networks and Systems, Part5: Communication Requirements for functions and device models, 1st ed, 2003.
- [6] K.-P. Brand and W. Wimmer, "The concept of IEC61850, A new approach for communication in substation automation and beyond," ABB Review, Special Report IEC61850, 2010.
- [7] H. Kirrmann, "Seamless Redundancy Bumpless Ethernet redundancy for substations with IEC61850," ABB Review, Special Report IEC61850, 2010.
- [8] IEEE. Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges. IEEE std 802.1D-2004.
- [9] IEC Standard. High-availability Automation Networks, Part 3: Parallel Redundancy Protocol (PRP) and High-availability Seamless Redundancy (HSR), IEC62439-3, The International Electrotechnical Commission. Geneva, Switzerland, 2010.
- [10] M. Shin and I. Joe, "Performance improvement for the HSR ring protocol with traffic control in smart grid," 4th International Conference-FGIT, vol.351, pp.48–55, Dec. 2012.
- [11] I.R. Abdulsalam and J.M. Rhee, "Improvement of High-Availability Seamless Redundancy (HSR) Unicast traffic Performance Using Port Locking," Proceedings of the 4th World Congress on Software Engineering-WCSE, HongKong, China, 3-4, pp.246–250, Dec. 2013.
- [12] S.A. Nsaif and J.M. Rhee, "Improvement of high-availability Seamless redundancy (HSR) traffic performance for smart grid communications," Journal of Communications and Networks, vol.14, no.6, pp.653–661, Dec. 2012.
- [13] S.A. Nsaif and J.M. Rhee, "RURT: A Novel Approach for Removing the Unnecessary Redundant traffic in Any HSR Closed-Loop Network Type," Proceedings of the International Conference on ICT Convergence 2013, Jeju, Korea, 14-16, pp.1003–1008, Oct. 2013.
- [14] S.A. Nsaif and J.-M. Rhee, "DVP: A Novel High-Availability Seamless Redundancy (HSR) Protocol Traffic-Reduction Algorithm for a Substation Automation System Network," Energies, vol.7, no.3, pp.1792–1810, March 2014.
- [15] S. Hong and I. Joe, "A Novel Packet Transmission Scheme with Different Periods According to the HSR Ring Direction in Smart Grid," FGIT 2012, vol.7709, pp.95–102, Dec. 2012.
- [16] M. Shin, M. Park, D. Oh, B. Kim, and J. Lee, "Clock synchronization for one-way delay measurement: a survey," CAN 2011, CCIS 199, vol.199, pp.1–10, 2011.
- [17] IEEE Std 1588-2008. Standard for a precision clock synchronization protocol for networked measurement and control systems.
- [18] OPNET [Online] <http://www.opnet.com/>
- [19] H.-S. Yang, S.-S. Kim, and H.-S. Jang, "Optimized Security Algorithm for IEC61850 based Power Utility System," Journal of Electrical Engineering and Technology, vol.7, no.3, pp.443–450, 2012

Seokjoon Hong received his the B.S. and M. S degrees in Department of Electronics and Computer Engineering from Hanyang University, Seoul, Korea in 2006, 2008 respectively. He is currently working toward Ph.D. degree in Computer & Software from Hanyang University, Seoul, Korea. His current research interests include 5G mobile communication, smart grid and redundant ethernet.

Ducsun Lim received his the B.S. degrees in Department of Computer Engineering from Hanyang cyber University, Seoul, Korea in 1998. From March 2014, he has M.S. degree in Computer & Software from Hanyang University, Seoul, Korea. His current research interests include SDN and Network Virtualization.

Inwhae Joe received his B.S. and M.S. degrees in Electronics Engineering from Hanyang University, Seoul, Korea, and his Ph.D. degree in Electrical and Computer Engineering from Georgia Institute of Technology, Atlanta, GA in 1998. Since 2002, he has been a faculty member in the Division of Computer Science & Engineering at Hanyang University, Seoul, Korea. His current research interests include mobile internet, cellular system and PCS, wireless ATM, mobile ad-hoc networks, multimedia networks.