ISSN: 2182-2069 / E-ISSN: 2182-2077

Optimizing Synchronization and Role-Based Access Control in Double-Surgeon Telesurgery Cockpits

Awwal Ishiaku[1*], and Christian Bassey[2]

[1*]Institute of Robotics and Computer Vision, Innopolis University, Innopolis, Russia.
a.ishiaku@innopolis.university, https://orcid.org/0009-0000-6360-3819

[2]Department of Security and Network Engineering, Innopolis University, Innopolis, Russia.
c.bassey@innopolis.university, https://orcid.org/0009-0001-1589-1374

Received: December 06, 2024; Revised: January 15, 2025; Accepted: January 30, 2025; Published: February 28, 2025

Abstract

The double-surgeon cockpit is an innovation in telesurgery that allows two surgeons to collaborate by remotely controlling a robot to perform surgical operations. However, it also introduces new challenges in synchronization and Role-Based Access Control (RBAC) due to the need for precise coordination and secure operation. In this paper, we address these challenges and propose an optimized synchronization strategy and implement RBAC within double-surgeon cockpits. The synchronization strategy combines passive latency monitoring and active command throttling to ensure coordinated actions from multiple surgeon consoles. Meanwhile, RBAC ensures that we assign specific permissions to each surgeon based on their roles, restricting access to only necessary robotic components, and preventing unauthorized actions. This dual approach aims to improve the operational efficiency, security, and overall success of telesurgical operations. The simulation results demonstrate the effectiveness of the proposed synchronization strategy, while a detailed RBAC framework ensures secure and efficient role management in telesurgery.

Keywords: Telesurgery, Double-Surgeon Cockpit, Latency, Synchronization, Role-Based Access Control.

1 Introduction

Telesurgery enables physicians to remotely control surgical robots across long distances to conduct surgery (Regina Chandra & Jayabal, 2019). This enables individuals in rural areas to obtain essential medical care (Özsoy & Alcan, 2017). Telesurgery has been realized since at least 2001, when Marescaux et al. conducted the first recorded telesurgery procedure. Telesurgery with robot-controlled surgical equipment improves the precision and accuracy of surgical interventions (Chatterjee et al., 2024). A recent advancement in telesurgery known as the "double-surgeon cockpit" (Oki et al., 2023) facilitates real-time collaboration between two surgeons from potentially distant locations. This approach combines the expertise of multiple surgeons to achieve better surgical outcomes (Sabah & Kaya, 2023).

Latency is a well-known problem in classical single-operator telesurgery procedures. We need the actions sent from the surgeon console to be delivered to the robot with minimal delay for the best surgical experience (Xu et al., 2014). The integration of multi-operator systems introduces new challenges in the area of synchronization and Role-Based Access Control (RBAC) (Craß et al., 2013; Boukerche et al.,

Journal of Internet Services and Information Security (JISIS), volume: 15, number: 1 (February), pp. 330-347.
DOI: *10.58346/JISIS.2025.I1.021*

*Corresponding author: Institute of Robotics and Computer Vision, Innopolis University, Innopolis, Russia.

2007; Pedersen, 2017). These challenges affect the safety, efficiency, and security of telesurgical procedures. The surgeons' consoles might have varying latencies, which introduces the need to ensure that commands sent from them to the robot are synchronized with each other to avoid accidents (Rayman et al., 2005; Rogers et al., 2017). In addition, we need to ensure that each surgeon has permission to access only the robotic component that they need for the surgical procedure (Mori et al., 2024).

This research aims to tackle these problems by concentrating on the optimization of synchronization in double-surgeon telesurgery cockpits. We investigate how to utilize adaptive synchronization and RBAC to improve the safety, efficiency, and security of multi-operator telesurgical systems. By doing so, we advance the overarching objective of enhancing telesurgery, rendering it a more feasible instrument for contemporary medicine.

Although prior studies have explored certain facets of single-operator telesurgery, limited research has comprehensively investigated the multi-operator dimension presented by the double-surgeon cockpit. In single-surgeon situations, synchronization generally employs buffering or latency offset techniques that fail to include concurrent control by numerous specialists. By contrast, our approach specifically tackles real-time concurrency and surgical collaboration when two surgeons concurrently send commands to the same robotic platform. This necessitates new mechanisms for conflict resolution, prioritization of commands, and split-second adjustments to accommodate varying latencies across different surgeon consoles.

Moreover, we extend beyond standard RBAC implementations often used in static enterprise systems by integrating dynamic, real-time permission checks tailored for critical telesurgery conditions (He et al., 2014). Traditional RBAC frameworks are not designed for the high-stakes environment of telesurgery, where immediate overrides and rapid changes in roles (for instance, from an assistant to a lead, or vice versa) may be required during emergencies. Our proposed RBAC configuration is thus novel in that it anticipates and responds to the concurrent input of two surgeons, granting or revoking access to specific robotic subsystems based on evolving surgical demands (Lee et al., 2022).

In summary, our approach introduces a synchronization strategy that adaptively manages latency variation between multiple surgeons in real time, combines it with an enhanced RBAC mechanism supporting instantaneous role changes and urgent override scenarios, and demonstrates through simulation and conceptual design how these elements together improve both the safety (by preventing conflicting robot commands) and the security (by preventing unauthorized actions) of a collaborative telesurgery environment. This work offers a practical framework for multi-operator telesurgery by addressing the distinct challenges of managing multiple surgeon consoles and ensuring real-time, role-specific access.

2 Background

Early telesurgery platforms (e.g., Marescaux et al., 2001) concentrated on single-surgeon control with stable connections, permitting minimal role switching or simultaneous inputs. In these systems, synchronization was simpler because a single operator issued commands, rendering buffering or latency compensation schemes typically sufficient. However, the advent of multi-surgeon arrangements creates a scenario in which two specialists, each assigned certain responsibilities, can issue commands simultaneously. Single-surgeon latency compensation is not sufficient to resolve conflicting command issues, as there is no built-in mechanism to determine which surgeon’s command should take precedence when timing overlaps.

Outside of medical contexts, multi-operator robot control has been explored in fields such as industrial automation, drone swarms, and collaborative assembly lines (Tian et al., 2022; Lee et al., 2020). While these studies address concurrency, they often assume lower stakes for short-term command collisions or provide hierarchical control (e.g., one "master" operator with priority, others in secondary roles). In double-surgeon telesurgery, both surgeons may be equally authoritative. A single "master surgeon" architecture could hamper the benefits of two experts actively collaborating and, in emergencies, hamper rapid overrides from either participant.

2.1. Synchronization in Double-Surgeon Telesurgery Cockpits

Coordination among many surgeons managing various facets of a telesurgery robot is essential (Wen et al., 2024). This is applicable to systems necessitating coordinated actions from several operators, particularly in telesurgery systems, where the consequences of failure are significant (Bouteraa & Ghommam, 2009; Anvari et al., 2005). Synchronization in this context entails synchronizing the efforts of the surgeon and harmonizing the temporal perception across the system components to guarantee seamless and secure procedures.

System Model and Clock Synchronization

The telesurgery system consists of two primary components: surgeon consoles and the robotic surgery platform (Mohan et al., 2021).

- **Surgeon Consoles:** These are the interfaces utilized by surgeons to engage with the telesurgery system. Surgeon consoles offer immediate input, manipulation of robotic equipment, and visualization of the operating area.
- **Robotic Surgery Platform:** This comprises the robotic arms, endoscopic cameras, and additional surgical instruments, serving as the fundamental infrastructure. These components convert the surgeon's actions into tangible movements within the patient's body (Menaka et al., 2022).

Each component functions on its local clock, which must be synced with a global reference time to guarantee the precise ordering and execution of time-stamped activities (Bassil et al., 2021).

Latency Variance

In addition to clock synchronization, we must minimize the latency differences between the physicians' consoles and the robotic surgical platform (Lum et al., 2009). Latency in this case, the duration between the time an action is performed by a surgeon and the time it is executed by the robot. Even with synchronized clocks, latency variations may cause desynchronization and impact the coordination of actions among surgeons. Significant latency delay begins around 300ms, and latency over 700ms is difficult to manage (Perez et al., 2016).

For example, if one surgeon's console shows a delay of 100ms and another shows a latency of 300ms, a 200ms discrepancy may lead to unintended consequences, such as:

- **Uncoordinated movements:** The robotic arms may operate asynchronously, potentially resulting in injury to tissue or equipment (Perez et al., 2016).
- **Misaligned time perception:** Surgeons may experience the same event at varying times, which may cause confusion and mistakes (Perez et al., 2016).

Telesurgery systems should measure, compensate for, and reduce latency variance to guarantee synchronized and coordinated execution of surgeons' actions.

Specific Synchronization Challenges

When two equally authorized surgeons issue overlapping commands (e.g., simultaneously moving robotic arms toward different targets), standard single-operator buffering methods cannot easily determine which action to prioritize. Command collisions may result in abrupt robot movements or errors if both commands arrive at nearly the same timestamp, making precise surgical maneuvers risky.

Even with universal time stamps, network inconsistencies mean a command sent by *Surgeon A* could arrive after *Surgeon B*'s later command. If the system naively executes commands by arrival time, Surgeon B's instructions might incorrectly override *Surgeon A*'s. In high-latency scenarios, this can lead to a race condition (Iorio et al., 2013) where neither surgeon has a clear understanding of the robot's immediate state.

In a collaborative setting, each surgeon needs the ability to override or interrupt the other if an urgent complication arises (Bardram et al., 2000). Yet, these interrupts themselves must be synchronized. If *Surgeon A* does not see *Surgeon B*'s override in time, they might continue with a conflicting maneuver. Effective synchronization must consider the latency between an "override request" and its acknowledgment by the system, guaranteeing secure transitions of control.

These problems highlight the necessity for adaptive monitoring, throttling methods, and real-time concurrency management.

Coordinating Real-Time Overrides

Some multi-operator systems allow a "priority shift" during emergencies (Bartek et al., 2019). For example, a senior user can override a junior user's inputs. But the logic is often based on set hierarchies, while many surgeries depend on two equally skilled surgeons to work together. Sometimes, one surgeon may need to override another if there is a complication or misunderstanding. To handle such override permissions, we need a synchronization protocol. This protocol will handle the latency and RBAC layer.

Synchronization Objective

Xu et al., (2014) have shown in their study that latency between 0 to 200 ms is good enough for surgeons. This has minimal impact on the performance of the surgery. They highlighted the different levels of acceptable delay and noted that a latency of 800 to 1000 ms is dangerous. With this understanding, developers can create systems that handle latency challenges more effectively.

Our goal is to create a framework that ensures that multi-surgeon systems are reliable even when there are latency challenges. We fix the synchronization challenges by using an adaptive synchronization strategy to detect when surgeon commands are not in sync with each other and to actively manage the latency variations.

2.2. Role-Based Access Control in Double-Surgeon Telesurgery Cockpits

Role-Based Access Control (RBAC) is a framework that limits access to system resources based on the user responsibilities (Sandhu, 1998). These methods ensure that users can only access the tools they need to do their jobs. RBAC is recognized for its function in controlling user access to information systems based on their jobs within the organization (Ferraiolo et al., 2001).

RBAC reduces the risk of unauthorized actions in telesurgery by making the roles and duties of each individual clear. With this method, some surgeons may be given the power to manage and monitor only necessary robotic components. The permissions assigned will depend on the responsibilities of each

member of the surgical team. This could stop users from performing unauthorized actions (Asif & Khondoker, 2020) that may endanger patient safety. This may also improve surgeon collaboration. The system can adjust to integrate new roles and permissions to correspond with medical innovation (Madugalla & Perera, 2024). For RBAC to work, you need to plan ahead, understand the surgical process, and ensure that permissions align with the requirements of each surgery and participant.

3 Proposed Adaptive Synchronization Strategy

A system that can switch between passive monitoring and active intervention is an effective solution because network delays often change. Our adaptive method can handle two surgeons working at the same time who may send overlapping commands. Traditional methods sometimes rely on uniform buffering or fixed latency offsets, which are not insufficient when two surgeons operate a robotic system. Our approach:

- **Combines Passive and Active Modes:** Instead of just tracking latency or uniformly buffering commands, we use passive monitoring that automatically switches to active throttling when a significant latency difference is found. This dual mode reduces delays and handles differences in real time better than just predicted or constant buffering methods.
- **Responds Dynamically to Varying Latencies:** In many current methodologies, one surgeon is named as the primary controller and the other limited control. Our architecture adjusts the rate of commands based on the real latency between the console and the robot. This lets surgeons collaborate without using a hierarchical control system.
- **Integrates a Safety Threshold:** By implementing an operational safety threshold, we ensure that the system halts when network conditions get too dangerous. This helps prevent significant robot error.
- This synchronization design is therefore novel in that it not only monitors and adjusts latency in real time but does so with explicit support for multiple active surgeons who may both initiate critical or overlapping commands. Such is not sufficiently addressed by earlier proposed multi-surgeon systems.

3.1. Passive Monitoring of Latency Variability

Latency Measurement and Collection

We propose to timestamp each action sent from a surgeon console with the local time at which it was sent. Upon receiving the action, the robotic platform logs the reception time. Formally, let $C = \{c_1, c_2, \ldots, c_n\}$ be the set of surgeon consoles. For each console $c_i \in C$, let $T = \{(t_{s,i,j}, t_{r,i,j}) \mid j \in N\}$ represent the set of timestamp pairs, where $t_{s,i,j}$ is the time when the j-th action is sent and $t_{r,i,j}$ is the time when it is received.

Latency Calculation

To find the latency for the j-th action from the console c_i, we include a clock synchronization offset Δ_i. The offset Δ_i is the difference between the sender's clock and the receiver's clock at the same instant. The latency for the j-th action is expressed in Eq. (1):

$$L_{i,j} = (t_{r,i,j} - t_{s,i,j}) + \Delta_i \quad (1)$$

This latency value shows how long it takes for an action to move from the surgeon's console to the robot.

Sliding Window Monitoring of Latency Differences

We propose to use a sliding window to monitor latency differences. This method involves calculating latency differences in a group of recent actions so that temporary spikes do not trigger false alarms. We define a sliding window W of size ω actions.

For each pair of consoles (c_i, c_k), we calculate the absolute latency difference for the j-th action as indicated in Eq. (2):

$$\Delta L_{i,k,j} = \left| L_{i,j} - L_{k,j} \right| \quad (2)$$

Where j includes the actions within the current window.

Threshold and Alert Mechanism

We propose to use a predefined threshold θ and an alert ratio α to determine when latency discrepancies are significant. The system triggers an alert if the ratio of latency differences within the window W that exceed θ is greater than α. An alert is triggered if the conditions in Eq. (3) are met:

$$\frac{1}{\omega} \sum_{j \in W} 1_{\{\Delta L_{i,k,j} > \theta\}} > \alpha \quad (3)$$

Where $1_{\{.\}}$ is the indicator function and it returns 1 if the condition inside the braces is true and 0 otherwise.

3.2. Active Command Throttling

In this case, command throttling controls the rate at which commands are sent based on real-time conditions to keep surgeon actions in sync. We suggest the integrating command throttling as a proactive measure to ensure that all surgeon consoles transmit commands with approximately the same latency, hence preserving synchronization.

Justification for Command Throttling

- **Minimizing Latency Impact:** Command throttling changes how often commands are sent based on real-time latency readings. By aligning the transmission rate to match the latency of the slowest console (identified as L_{max}), commands from other consoles are sent more slowly. This reduces the possibility of asynchronous actions during surgical procedures.
- **Safety and Reliability:** Instead of adding latency with buffering systems, command throttling enhances command delivery while maintaining real-time responsiveness. Buffering could negatively impact command execution which may compromise the accuracy of surgery and the patient's safety. Additionally, predictive measures might work in theory, but they might make telesurgery less safe because of the need for precise and quick control.
- **Operational Efficiency:** Command throttling improves operational efficiency by changing the rate of command transmission based on latency assessments. This ensures that network resources are used optimally while preserving synchronization between the surgeon console and the robotic.

Throttling Strategy

The goal is to adjust the command transmission rate from each console based on real-time latency measurements. Let τ_i represent the transmission rate adjustment factor for console c_i. The transmission rate R_i for the console c_i is adjusted dynamically, as shown in Eq. (4):

$$R_i = R_{base} \,.\, \tau_i \quad (4)$$

Where R_{base} is the base transmission rate, and τ_i is determined in Eq. (5) as the ratio of the maximum observed latency L_{max} to the current latency $L_{i,j}$:

$$\tau_i = \frac{L_{max}}{L_{i,j}} \qquad (5)$$

This adjustment guarantees that the transmission rate R_i for each console is proportional to its latency, thereby aligning all consoles with the latency of the slowest console.

Command Queue and Delay Mechanism

The commands from each surgical console are queued for a predetermined duration as a result of the throttling mechanism. Let $D_{i,j}$ represent the delay time for the j-th command from the console c_i. The delay time is computed in Eq. (6) to align the command transmission timing with the synchronized rate:

$$D_{i,j} = \left(\frac{L_{max}}{L_{i,j}} - 1\right).T \qquad (6)$$

where T is the base delay period. During this delay, commands are held in the queue on the surgeon console, ensuring that transmission to the robotic platform is throttled according to the calculated rate.

Implementation Considerations

Implementing the throttling method requires ongoing observation of latency fluctuations and the adaptive modification of transmission rates. The command queue and delay mechanism provide synchronized transmission of all commands, hence reducing latency disparities.

3.3. Simulation of Active Command Throttling

We assessed the delay of command transfers from three simulated surgeon consoles to a simulated robotic system. The consoles, identified as console_1, console_2, and console_3, were evaluated to assess and analyze the latency in command transmission. Each console transmitted a total of 5000 commands at a frequency of 5 instructions per second.

Unthrottled Command Latency

Figure 1 depicts the latency recorded for each console in the absence of command throttling. As anticipated, console_1, operating on the same host as the robot, demonstrated little delay. Console_2 had moderate delay, and console_3 demonstrated the highest latency of the three consoles.

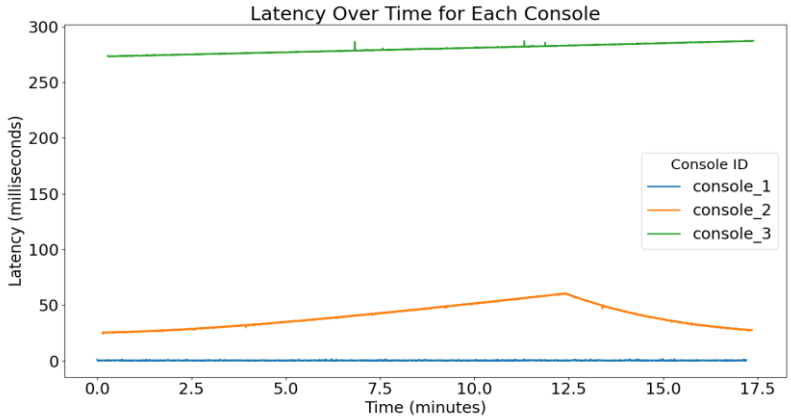

Figure 1: Latency over time for each console without throttling

Throttled Command Latency

To mitigate the disparate latencies and synchronize orders from all consoles, we re-executed the simulation with command throttling activated. The objective was to synchronize the command execution timings of all consoles with the console exhibiting the highest delay. The delay threshold has been established at 50 milliseconds. Figure 2 displays the outcomes subsequent to the implementation of the throttling mechanism. The latencies of console_1 and console_2 were augmented to enhance synchronization with console_3, illustrating the efficacy of the throttling mechanism in aligning command delivery.

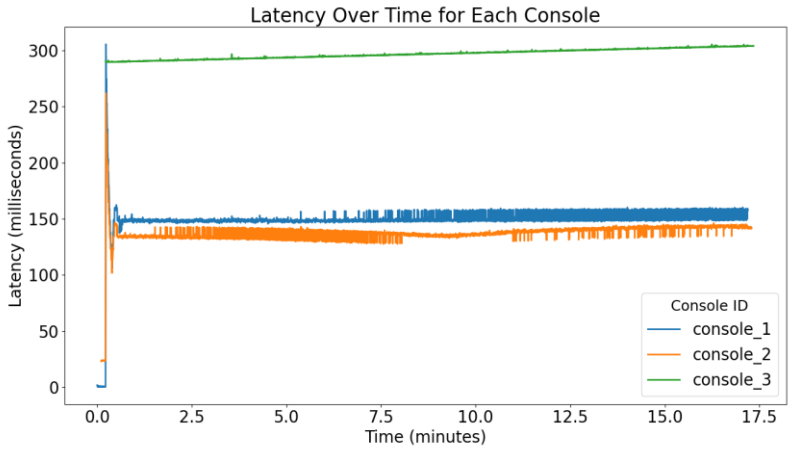

Figure 2: Latency over time for each console with throttling

Discussion

The data indicate that, in the absence of throttling, there exists a considerable disparity in latency among the three consoles, with console_3 consistently exhibiting the highest latency. The implementation of the throttling mechanism enhanced the synchronization of command delivery times.

3.4. Operational Safety Threshold

Exceeding a preset safety threshold in latency disparities between surgeon consoles in telesurgery systems signifies potential synchronization problems that may jeopardize patient safety and surgical accuracy. The safety threshold is the latency variance beyond which active command throttling fails to synchronize surgeons' actions. As soon as these differences are detected, operations must stop immediately to prevent unwanted robotic movements and ensure that surgical procedures stay accurate and in sync across all consoles.

Safety Threshold Implementation

To implement an operational safety threshold, we need the following:

1. Threshold Definition: Establish a safety threshold $\Delta L_{threshold}$ over which operations are considered dangerous. This threshold is determined by system requirements and safety regulations. A safety response is triggered if the latency difference exceeds $\Delta L_{threshold}$.
2. Safety Response Mechanism: Establish a protocol to stop activities when $\Delta L_{i,j}$, the latency difference between actions from consoles c_i and c_j, exceeds $\Delta L_{threshold}$. This could involve:

- Sending an instant notification to operators and surgeons.
- Automatically halting the robotic platform and surgical procedures.
- Commencing procedures for system diagnostics and recalibration.

4 Implementing Role-Based Access Control

In the following sections, we provide more details about the definition of permissions for robotic components, the pre-surgery planning and role assignment process, as well as the challenges and solutions related to the implementation of a role-based access control system in telesurgery.

4.1. Defining Permissions for Robotic Components

The level of control in a telesurgery system is important to its success. This brings the need for a well-organized permission system for surgeons to control robotic components. Each surgeon will be able to perform the actions they are permitted to. RBAC systems will categorize permissions into various types depending on the needed robotic operation and control.

Table 1 outlines the permissions we derived from (Laso et al., 2019) that are relevant to telesurgery systems. It serves as a framework to help allocate access levels to robotic components. Each type of permission is designed to maintain operational security and flexibility. Surgeons must have only the necessary access to perform their tasks, and the system must prevent unauthorized actions. The allocation of these permissions is an important aspect of pre-surgery preparation. It should be done after considering the requirements of the operation, the robotic components that will be used, and the roles of each member of the team.

Table 1: Permissions in Role-Based Telesurgery Systems

Permission Type	Description
Control	Allows a surgeon to directly manipulate a robotic component, such as moving a robotic arm or activating an endoscopic camera.
Adjustment	Permits a surgeon to modify the settings or parameters of a robotic component. This allows adjusting aspects such as speed, range of motion or tool selection.
Observation	Grants a surgeon the ability to view the status, output, and live feed of a robotic component without allowing direct control or adjustments.
Override	Provides the capability to temporarily take control over a robotic component. This is usually reserved for senior surgeons or specialists who respond to emergency situations during surgery.
Lock/Unlock	Enables a surgeon to lock a robotic component and subsequently unlock it. This is important for patient safety during the critical phases of surgery.

The assigned permissions must be planned appropriately and allow the surgeon to contribute to the operation. This method of permission assignment enhances the safety of operations. It also guarantees that the system can adapt to the dynamic nature of surgical procedures.

4.2. Pre-Surgery Planning and Role Assignment

Presurgical planning is the process of organizing and aligning resources, staff, and protocols before a surgical procedure to improve patient safety and outcomes (Teunissen et al., 2020). The responsibilities of each surgeon are defined during the planning phase. With the introduction of RBAC, we need to consider the following workflow before surgery commences:

- **Surgical Team Assembly:** Identify and assemble the surgical team, including all surgeons who will participate in the procedure.
- **Role Definition:** Based on the surgery's requirements and each surgeon's expertise, define specific roles for the procedure.
- **Permission Assignment:** Assign permissions to each role, determining what each surgeon can control, adjust, observe, or override in the robotic system.
- **Operational Protocols:** Establish protocols for emergencies, including how the override permission should be used.

Defining Roles and Permissions

The use case outlined in Listing 1 demonstrates how roles and permissions are programmatically assigned to surgeons and anesthesiologists. This allows each participant to interact with the system within their areas of expertise and responsibility.

Listing 1 Defining roles and permissions

```
roles = {Lead_Surgeon, Assistant_Surgeon, Anesthesiologist}
permissions = {Control, Adjustment, Observation, Override, Lock/Unlock}

rolePermissions = {
  Lead_Surgeon: [Control, Adjustment, Observation, Override, Lock/Unlock],
  Assistant_Surgeon: [Control, Observation],
  Anesthesiologist: [Control, Adjustment, Observation]
}
```

This structure allows for a flexible and secure assignment of operational capabilities and grants each team member the access needed to fulfill their designated role.

Assigning Robotic Components to Permissions

Once roles and permissions are defined, they must be mapped to particular robotic components. As shown in Listing 2, this mapping specifies who can do what on every component of the robot.

Listing 2 Assigning robotic components to permissions

```
roboticComponents = {Robotic_Arm, Endoscope, Anesthesia_Machine, Vital_Signs_Monitor}

componentPermissions = {
  Lead_Surgeon: {
    Robotic_Arm: [Control, Adjustment, Lock/Unlock, Override],
    Endoscope: [Control, Observation]
    Vital_Signs_Monitor: [Observation]
  },
  Assistant_Surgeon: {
    Endoscope: [Control, Observation]
    Vital_Signs_Monitor: [Observation]
  },
  Anesthesiologist: {
    Endoscope: [Observation]
    Anesthesia_Machine: [Control, Adjustment],
```

```
    Vital_Signs_Monitor: [Observation]
  }
}
```

Each robotic component has a particular function in the surgical ecosystem.

- **Robotic Arm:** Used for the manipulation of surgical instruments and tools within the operating area.
- **Endoscope:** Enables the visualization of internal structures and organs during procedures.
- **Anesthesia Machine:** Administers anesthesia to the patient for comfort and pain management during the surgery.
- **Vital Signs Monitor:** Monitors the patient's vital signs, including heart rate, blood pressure, and oxygen levels for feedback on the patient's condition.

The system ensures that surgeons' interactions are aligned with their responsibilities by assigning permissions based on their roles.

After roles and permissions have been assigned, the surgical team establishes protocols for emergency situations and any possible overrides. Such protocols depend on the specifics of surgery and the regulatory framework that guides medical practice.

4.3. Real-Time Role Shifts, Concurrency-Aware Permissions, and Immediate Overrides

We add extra features to the traditional RBAC model to allow changes to roles and permissions during live procedures. Specifically, we incorporate real-time role shifts, concurrency-aware permissions, and immediate overrides. These features are necessary in the double-surgeon cockpit where two equally competent surgeons may access robotic components at the same time.

Real-Time Role Shifts

A real-time role shift is the instantaneous assignment (or revocation) of a role $r \in R$ to a surgeon u while the telesurgery system is operational, without requiring a system restart or manual reconfiguration step.

- **Role Set and Assignment Function:** Let $R = \{r_1, r_2, r_3, \ldots, r_m\}$ be the set of all possible roles in the telesurgery system (e.g., *Lead_Surgeon*, *Assistant_Surgeon*, *Trainee*,…).

  We denote the set of surgeons by $U = \{u_1, u_2, u_3, \ldots, u_n\}$.

  A function $RolesAssigned: U \rightarrow 2^R$ maps each surgeon to a subset of roles they currently hold.
- **Real-Time Update Operation:** We define a function $AssignRole(u, r)$ that updates $RolesAssigned(u)$ by adding r to the surgeon u's current roles; similarly $RevokeRole(u, r)$ removes r.
- These operations can occur mid-procedure (e.g., if a senior surgeon delegates a specialized tool to another surgeon).
- Each update triggers an event to recalculate permissible actions (Section 4.3.2) in near real time.

1. **State Machine View:** Conceptually, each surgeon u is in a role state $S(u) \subseteq R$. We can model real-time role shifts as a finite state machine. Eq. (7) denotes role assignment $AssignRole(u, r)$, and Eq. (8) denotes role revocation $RevokeRole(u, r)$.

$$S(u) \rightarrow S(u) \cup \{r\} \quad (7)$$

$$S(u) \rightarrow S(u) \backslash \{r\} \quad (8)$$

These transitions must be validated against safety constraints, ensuring the requested role shift does not conflict with concurrency rules or emergency lockouts.

Concurrency-Aware Permissions

Concurrency-aware permissions extend the RBAC concept to account for multiple surgeons requesting simultaneous access to the same robotic components, each with distinct privileges or real-time roles.

1. **Permission Sets and Resource Mapping:** Let P be the universal set of permissions (e.g., *Control*, *Adjustment*, *Observation*, etc.).

   Each role $r \in R$ maps to a subset of permissions $Perm(r) \subseteq P$

   For each robotic component $c \in C$, there is a function $Allowed(r, c) \subseteq P$ that specifies which permissions are valid for role r on component c.

- For example, a Lead_Surgeon might have:
$$Allowed(Lead\_Surgeon, Robotic\_Arm) = \{Control, Adjustment, Override\}$$
- An Assistant_Surgeon might have:
$$Allowed(Assistant\_Surgeon, Endoscope) = \{Control, Observation\}$$

2. **Concurrent Access Logic:** When two surgeons u_i and u_j attempt actions on the same robotic component c, the concurrency-aware system must reconcile possible overlapping permissions. We define a function $ConcurrentPerms(u_i, u_j, c)$ in Eq. (9) to yield the effective permissions if both surgeons simultaneously operate component c.
$$ConcurrentPerms(u_i, u_j, c) = Allowed(r_i, c) \cup Allowed(r_j, c) \quad (9)$$
   where $r_i \in RolesAssigned(u_i), r_j \in RolesAssigned(u_j)$

   However, if a policy states that only one user can hold *Control* at a time, the concurrency manager must apply a conflict resolution step. This may degrade *Control* for the second user to *Observation*, or put the second user in a wait state, depending on the configured policy.

3. **Conflict Resolution:** To prevent collisions, concurrency-aware permissions often rely on a priority or arbitration mechanism. One approach is to define a total ordering $\preccurlyeq$ over roles (e.g., *Lead_Surgeon* $\preccurlyeq$ *Assistant_Surgeon*, indicates the lead surgeon outranks the assistant in case of conflict).

   Another approach is to monitor latency or the command queue to decide which user's command is enacted first. The final set of effective permissions is shown in Eq. (10):
$$EffectivePerms(u_i, c) = Resolve(Allowed(RolesAssigned(u_i), c), ConcurrentPerms(u_i, u_j, c), \dots) \quad (10)$$
   Where $Resolve(.)$ applies concurrency rules (priority, queue time, or policy constraints) in near real time.

Immediate Overrides

An immediate override is a high-priority operation that escalates a surgeon's permissions or revokes certain permissions from another user to manage emergencies or critical actions.

1. **Override Role r_O:** We define a special role $r_O \in R$ (e.g., *Emergency_Override*) that contains a superset of necessary permissions across components to handle urgent scenarios, such as controlling the robotic arm, endoscope, anesthesia settings, etc. A user who acquires r_O can forcibly override conflicting commands from other surgeons.
2. **Trigger Condition:** The override can be triggered by manual request from a surgeon, or by the system when an anomaly is detected due to high and varied latency, concurrency checks, or other unsafe conditions. The system may automatically elevate the user with the most stable connection to r_O.

3. **Override Safety:** An override should not compromise patient safety. Therefore, it may have built-in checks, such as:
   - Confirming the requesting surgeon's credentials (i.e., they must already be assigned a high-level role, like *Lead_Surgeon*).
   - Ensuring the system is not already in a halted state due to extreme latency or an active procedure pause.
   - Logging and possibly requiring secondary confirmation if local regulations demand a "two-physician confirmation" process for certain overrides.

Integration With Synchronization Logic

These real-time RBAC features (role shifting, concurrency rules, immediate overrides) also interface with the adaptive synchronization layer. If the system detects that one surgeon's connection becomes dangerously latent, the concurrency manager can:

- Temporarily revoke or downgrade that surgeon's role, reducing the risk of collision with commands that arrive far out of sync.
- Grant *Override* to the surgeon with stable latency if an urgent decision or a critical command queue is building.

This integration ensures that latency management (from the synchronization subsystem) and permission management (from the RBAC subsystem) coordinate to maintain safety and avoid the robotic conflicts that can arise when two surgeons issue commands concurrently under unpredictable network conditions.

5 Discussion

Although our adaptive synchronization and RBAC mechanisms have been demonstrated through simulation, successfully deploying them in a hospital environment entails additional design and integration work. Hospitals commonly rely on virtual private networks (VPNs) or dedicated telemedicine links, which can introduce variable bandwidth and transient latency spikes. Our system's passive monitoring and command throttling approach is intended to dynamically adjust command rates in response to these spikes. However, real-world usage may require failover strategies. For example, quickly switching to a local backup or buffering system when remote connectivity degrades below an acceptable threshold.

Another important consideration is resource utilization. Complex telesurgery procedures can generate large volumes of real-time data (e.g., high-definition video feeds, multiple control streams, haptic feedback). While our simulations indicate that the overhead of continuous latency checks and on-the-fly RBAC validation is manageable, more thorough testing under real hospital workloads will be needed to confirm that the approach does not overburden network or computing resources.

5.1. Addressing Practical Network Constraints

Many hospitals use segmented networks with strict firewall rules and bandwidth allocation, especially for operating rooms. These security measures can influence end-to-end latency and packet arrival consistency. To mitigate these issues, our methodology includes:

- Adaptive throttling to handle periods of low throughput or high jitter, by slowing down high-frequency command bursts.

- Operational safety thresholds that halt procedures if latency variance escalates beyond a safe level, preventing the risk of out-of-sync or conflicting commands under severely unstable network conditions.

Before clinical use, we recommend pilot connectivity tests and controlled “network stress” simulations to verify that hospital IT policies and bandwidth constraints do not compromise the safety or speed of remote surgical operations.

5.2. Integration into Clinical Workflow

A double-surgeon cockpit fundamentally changes how surgical teams operate, so user training is essential. Both surgeons must become accustomed to how the system:

- Evens out timing differences between consoles, sometimes adding brief delays or queued commands.
- Handles real-time overrides, so that one surgeon’s emergency intervention for example does not surprise the other.
- Manages concurrent input by enforcing role-based permissions that can change during the procedure.

Preparation would involve simulation-based training, like using existing robotic surgery simulators where both surgeons practice collaborative procedures under varying simulated network conditions. This procedure can make them more familiar with system functions including UI alerts that indicate the system is actively throttling commands or granting an override request.

5.3. Regulatory Compliance

Since telesurgery systems are considered medical devices, they need to comply with certain standards and guidelines. We must maintain an audit trail of actions related to RBAC and adaptive synchronization for post-operative review and traceability. It should have a risk management strategy that is aligned with standards like ISO 14971 to identify, evaluate, and mitigate hazards such as extreme latency spikes or unauthorized overrides. There is a fail-safe feature in the suggested framework that causes a "safe pause" if certain conditions are met. All of these steps work together to make sure that patient safety is top priority.

5.4. Future Plans for Validation and Deployment

We expect several important phases to verify that our suggested approach can efficiently go from simulation to clinical reality. First, we need to perform extended simulation under realistic conditions. This includes using network topologies that are similar to those in hospitals, encryption methods, and audio and video streaming. These will help validate that command throttling and latency monitoring function reliably under near-real conditions.

Formal studies of usefulness will also be very important. By interviewing surgeons, surgical residents, and operating room staff, we can find out how well the concurrency management and role-based rights are received by users. Getting feedback on the design of the interface and the override processes will help improve the system.

In addition to usability and simulation studies, pilot telesurgery reviews need to be done. Subject to institutional ethics permissions, they will comprise test techniques on animal models or cadaver labs. The system's performance will be judged during real surgeries, with a focus on operation time, error rates, success in timely overrides, and any unexpected system halts.

We must consider the human factor of double-surgeon telesurgery. Simulation-based training courses will help familiarize surgeons with concurrency mechanisms. We need well-designed user interfaces that use color-coded status indicators and also shared action logs to help increase cooperation. If the system is not well-understood, surgeons may provide conflicting commands or miss important alarms. Communication protocols, role definitions, and interface signals must account for real-world surgeon behavior to help reduce errors. Although at this point we are not using particular user-centered designs, the success of any multi-operator telesurgery system depends on how well it combines technical implementations with human behavior.

Finally, a regulatory and compliance review is necessary. We can address safety and data privacy concerns by collaborating with relevant authorities. We can ensure that our proposed method is not merely a theoretical framework by planning these next phases. It must be noted that the framework can only be ready for practical adoption once rigorous technical and clinical milestones are achieved.

6 Conclusion

Our passive monitoring feature detects when there is excessive latency discrepancy between surgeons who are simultaneously controlling a robot. The active command throttling feature ensures that the latency difference between the surgeons is approximately the same if the discrepancy is minor. It achieves this by queuing commands and releasing them at an adjusted rate. To prevent conflict, the throttling mechanism gives priority to the console with higher latency compensation if surgeons are sending high frequency commands at the same time. The system triggers a temporary halt to avoid unsafe simultaneous movements if latency differences exceed the operational safety threshold.

In a high-latency environment, *Command A* from *Surgeon 1* can arrive after *Command B* from *Surgeon 2*, even if *Surgeon 1* initiated it first. By time-stamping commands and comparing send/receive times, the system can reorder actions internally or apply brief holds to realign out-of-sequence instructions. Throttling also helps smooth out arrival disparities by synchronizing the rate of outgoing messages, thus mitigating race conditions in which commands arrive in a confusing order.

Occasionally, one surgeon needs to override or interrupt the other (e.g., to address a sudden complication). Our adaptive synchronization coordinates the override commands just like other high-priority inputs. First, any large latency disparity would trigger the safety threshold check, ensuring both surgeons see the robot's most recent state before an override takes full effect. Next, by integrating with RBAC, the system grants immediate override permissions to the requesting surgeon if policy permits. During the override, throttling remains active, preventing the other surgeon's delayed commands from inadvertently executing after the override is in place.

This mapping of concurrency pitfalls (conflicting commands, out-of-order arrivals, and parallel overrides) to the passive monitoring, active throttling, and safety threshold mechanisms illustrates how our synchronization strategy handles real-time multi-operator control. We ensure that surgeons can collaborate without risking collisions or out-of-sync manipulations by combining these components.

The proposed implementation of RBAC in double-surgeon telesurgery cockpits enhances the security of telesurgical procedures. The system ensures that each surgeon interacts with the robotic components within their designated operational boundaries. This approach reduces the risk of unauthorized actions and also creates an environment where each team member can perform their tasks.

In summary, our work introduces an adaptive synchronization strategy and a dynamic RBAC mechanism that address the challenges of multi-surgeon telesurgery. Previous methods usually relied on a single operator or a strict hierarchy. We take a different approach by managing latency in real time

and continuously checking each surgeon's role. We integrate passive monitoring, command throttling, and on-the-fly override permissions. This creates a framework that handles conflicting commands, out-of-order arrivals, and urgent role changes. This framework surpasses standard telesurgery models and it shows that we can combine real-time concurrency and access control to improve patient safety.

References

[1] Anvari, M., Broderick, T., Stein, H., Chapman, T., Ghodoussi, M., Birch, D. W., ... & Goldsmith, C. H. (2005). The impact of latency on surgical precision and task completion during robotic-assisted remote telepresence surgery. *Computer Aided Surgery*, *10*(2), 93-99. https://doi.org/10.3109/10929080500228654

[2] Asif, M. R. A., & Khondoker, R. (2020). Cyber Security Threat Modeling of A Telesurgery System. *In 2020 2nd International Conference on Sustainable Technologies for Industry* (Vol. 4, pp. 1-6).

[3] Bardram, J. E. (2000). Temporal coordination–on time and coordination of collaborative activities at a surgical department. *Computer Supported Cooperative Work (CSCW)*, *9*, 157-187. https://doi.org/10.1023/A:1008748724225

[4] Bartek, M. A., Saxena, R. C., Solomon, S., Fong, C. T., Behara, L. D., Venigandla, R., ... & Nair, B. G. (2019). Improving operating room efficiency: machine learning approach to predict case-time duration. *Journal of the American College of Surgeons*, *229*(4), 346-354. https://doi.org/10.1016/j.jamcollsurg.2019.05.029

[5] Bassil, J., Piranda, B., Makhoul, A., & Bourgeois, J. (2021, November). Enhanced Precision Time Synchronization for Modular Robots. In *2021 IEEE 20th International Symposium on Network Computing and Applications (NCA)* (pp. 1-5). IEEE. https://doi.org/10.1109/NCA53618.2021.9685103

[6] Boukerche, A., Shirmohammadi, S., & Hossain, A. (2007). Prediction-based decorators for distributed collaborative haptic virtual environments. *International journal of computer applications in technology*, *29*(1), 81-88. https://doi.org/10.1504/IJCAT.2007.014063

[7] Bouteraa, Y., & Ghommam, J. (2009, March). Synchronization control of multiple robots manipulators. In *2009 6th International Multi-Conference on Systems, Signals and Devices* (pp. 1-6). IEEE. https://doi.org/10.1109/SSD.2009.4956742

[8] Chatterjee, S., Das, S., Ganguly, K., & Mandal, D. (2024). Advancements in robotic surgery: innovations, challenges and future prospects. *Journal of Robotic Surgery*, *18*(1), 28. https://doi.org/10.1007/s11701-023-01801-w

[9] Craß, S., Donz, T., Joskowicz, G., Kuhn, E., & Marek, A. (2013). Securing a space-based service architecture with coordination-driven access control. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, *4*(1), 76-97.

[10] Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., & Chandramouli, R. (2001). Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, *4*(3), 224-274. https://doi.org/10.1145/501978.501980

[11] He, W., Ni, S., Chen, G., Jiang, X., & Zheng, B. (2014). The composition of surgical teams in the operating room and its impact on surgical team performance in China. *Surgical endoscopy*, *28*, 1473-1478. https://doi.org/10.1007/s00464-013-3318-4

[12] Iorio, M., Risso, F., & Casetti, C. (2021). When latency matters: measurements and lessons learned. *ACM SIGCOMM Computer Communication Review*, *51*(4), 2-13. https://doi.org/10.1145/3503954.3503956

[13] Laso, P. M., Brosset, D., & Giraud, M. A. (2019, June). Defining role-based access control for a secure platform of unmanned surface vehicle fleets. In *OCEANS 2019-Marseille* (pp. 1-4). IEEE. https://doi.org/10.1109/OCEANSE.2019.8867389

[14] Lee, H., Enriquez, J. L., & Lee, G. (2022). Robotics 4.0: Challenges and Opportunities in the 4th Industrial Revolution. *Journal of Internet Services and Information Security*, *12*(4), 39-55. https://doi.org/10.58346/JISIS.2022.I4.003

[15] Lee, H., Liau, Y. Y., Kim, S., & Ryu, K. (2020). Model-based human robot collaboration system for small batch Assembly with a virtual fence. *International journal of precision Engineering and manufacturing-green technology*, *7*, 609-623. https://doi.org/10.1007/s40684-020-00214-6

[16] Lum, M. J., Rosen, J., King, H., Friedman, D. C., Lendvay, T. S., Wright, A. S., ... & Hannaford, B. (2009, September). Teleoperation in surgical robotics–network latency effects on surgical performance. In *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (pp. 6860-6863). IEEE. https://doi.org/10.1109/IEMBS.2009.5333120

[17] Madugalla, A. K., & Perera, M. (2024). Innovative uses of medical embedded systems in healthcare. *Progress in Electronics and Communication Engineering*, *2*(1), 48–59. https://doi.org/10.31838/PECE/02.01.05

[18] Marescaux, J., Leroy, J., Gagner, M., Rubino, F., Mutter, D., Vix, M., ... & Smith, M. K. (2001). Transatlantic robot-assisted telesurgery. *Nature*, *413*(6854), 379-380. https://doi.org/10.1038/35096636

[19] Menaka, S. R., Gokul Raj, M., Elakiya Selvan, P., Tharani Kumar, G., & Yashika, M. (2022). A Sensor based Data Analytics for Patient Monitoring Using Data Mining. *International Academic Journal of Innovative Research*, *9*(1), 28–36. https://doi.org/10.9756/IAJIR/V9I1/IAJIR0905

[20] Mohan, A., Wara, U. U., Shaikh, M. T. A., Rahman, R. M., Zaidi, Z. A., & Shaikh, M. T. A. (2021). Telesurgery and robotics: an improved and efficient era. *Cureus*, *13*(3). https://doi.org/10.7759/cureus.14124

[21] Mori, M., Hirano, S., Hakamada, K., Oki, E., Urushidani, S., Uyama, I., ... & Morohashi, H. (2024). Clinical practice guidelines for telesurgery 2022: Committee for the promotion of remote surgery implementation, Japan Surgical Society. *Surgery today*, *54*(8), 817-828. https://doi.org/10.1007/s00595-024-02863-5

[22] Oki, E., Ota, M., Nakanoko, T., Tanaka, Y., Toyota, S., Hu, Q., ... & Mori, M. (2023). Telesurgery and telesurgical support using a double-surgeon cockpit system allowing manipulation from two locations. *Surgical endoscopy*, *37*(8), 6071-6078. https://doi.org/10.1007/s00464-023-10061-6

[23] Özsoy, O., & Alcan, S. (2017). Horizontal Equity in Delivery of Health Care in Turkey. *International Academic Journal of Economics*, *4*(2), 40–61.

[24] Pedersen, R. D. (2017). Two-Person Control: a brief history and modern industry practices. https://doi.org/10.2172/1374246

[25] Perez, M., Xu, S., Chauhan, S., Tanaka, A., Simpson, K., Abdul-Muhsin, H., & Smith, R. (2016). Impact of delay on telesurgical performance: study on the robotic simulator dV-Trainer. *International journal of computer assisted radiology and surgery*, *11*, 581-587. https://doi.org/10.1007/s11548-015-1306-y

[26] Rayman, R., Primak, S., Patel, R., Moallem, M., Morady, R., Tavakoli, M., ... & Croome, K. (2005). Effects of latency on telesurgery: an experimental study. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2005: 8th International Conference, Palm Springs, CA, USA, October 26-29, 2005, Proceedings, Part II 8* (pp. 57-64). Springer Berlin Heidelberg. https://doi.org/10.1007/11566489_8

[27] Regina Chandra, W., & Jayabal, R. (2019). Scientometric Study of the Indian Journal of Plastic Surgery. *Indian Journal of Information Sources and Services*, *9*(2), 81–84. https://doi.org/10.51983/ijiss.2019.9.2.618

[28] Rogers, H., Khasawneh, A., Bertrand, J., & Madathil, K. C. (2017, September). An investigation of the effect of latency on the operator's trust and performance for manual multi-robot teleoperated tasks. In *Proceedings of the Human Factors and Ergonomics Society Annual*

Meeting (Vol. 61, No. 1, pp. 390-394). Sage CA: Los Angeles, CA: SAGE Publications. https://doi.org/10.1177/1541931213601579

[29] Sabah, I. F., & Kaya, G. A. (2023). A Study on the Experience of Online Learning in Light of the Corona Pandemic (COVID-19). *International Journal of Advances in Engineering and Emerging Technology*, *14*(1), 1–7.

[30] Sandhu, R. S. (1998). Role-based access control. In *Advances in computers*, *46*, 237-286. Elsevier.

[31] Teunissen, C., Burrell, B., & Maskill, V. (2020). Effective surgical teams: an integrative literature review. *Western journal of nursing research*, *42*(1), 61-75. https://doi.org/10.1177/0193945919834896

[32] Tian, Y., Chang, Y., Arias, F. H., Nieto-Granda, C., How, J. P., & Carlone, L. (2022). Kimera-multi: Robust, distributed, dense metric-semantic slam for multi-robot systems. *IEEE Transactions on Robotics*, *38*(4). https://doi.org/10.1109/TRO.2021.3137751

[33] Wen, X., Wang, Y., Zheng, X., Wang, K., Xu, C., & Gao, F. (2024, May). Simultaneous Time Synchronization and Mutual Localization for Multi-robot System. In *2024 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 2603-2609). IEEE. https://doi.org/10.1109/ICRA57147.2024.10610915

[34] Xu, S., Perez, M., Yang, K., Perrenot, C., Felblinger, J., & Hubert, J. (2014). Determination of the latency effects on surgical performance and the acceptable latency levels in telesurgery using the dV-Trainer® simulator. *Surgical endoscopy*, *28*, 2569-2576. https://doi.org/10.1007/s00464-014-3504-z

Authors Biography

Awwal Ishiaku is currently pursuing a Ph.D. degree at Innopolis University. His main research interests include cybersecurity, telesurgery robots, and cloud & virtualization technologies.

Christian Bassey is a security engineer whose research interests focuses on the security of cyber physical systems. He has a masters degree in security and network engineering from Innopolis University.