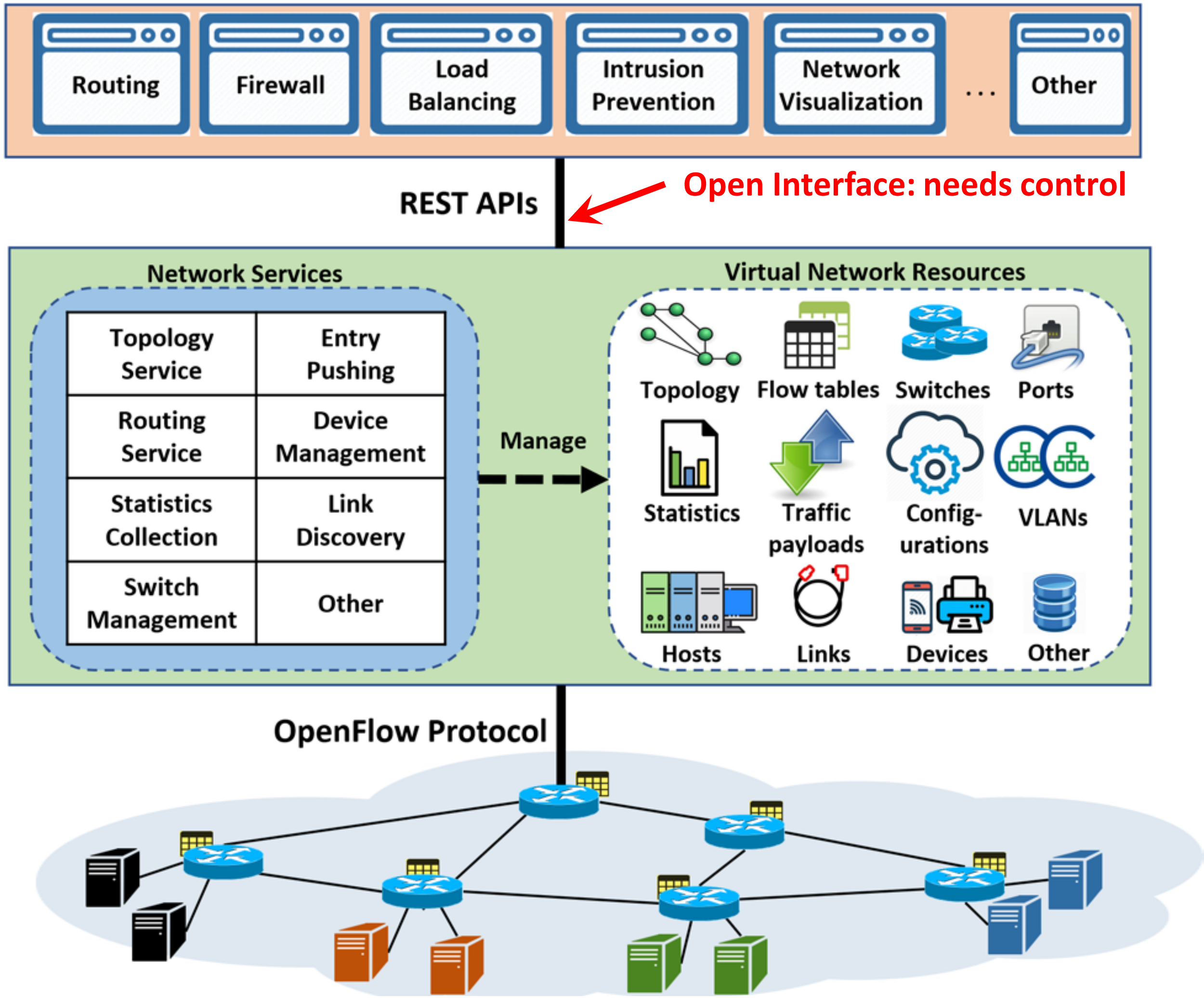
Access Control for Software Defined Networks: (SDN-RBAC Model)

Lecture 9-2

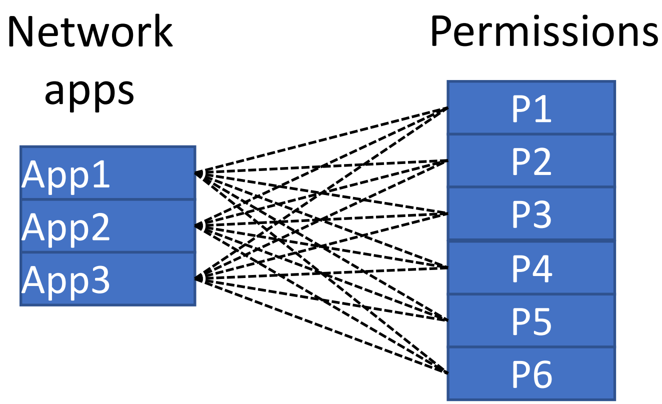
Access Control in SDN

- Control which subjects (network apps) can access which objects (virtual network resources) for performing which actions (SDN operations).



Access Control with capability-based approaches

- Capability-based approaches
  - Direct relation between operations and apps.
  - Well studied and known to have administrative complexities.

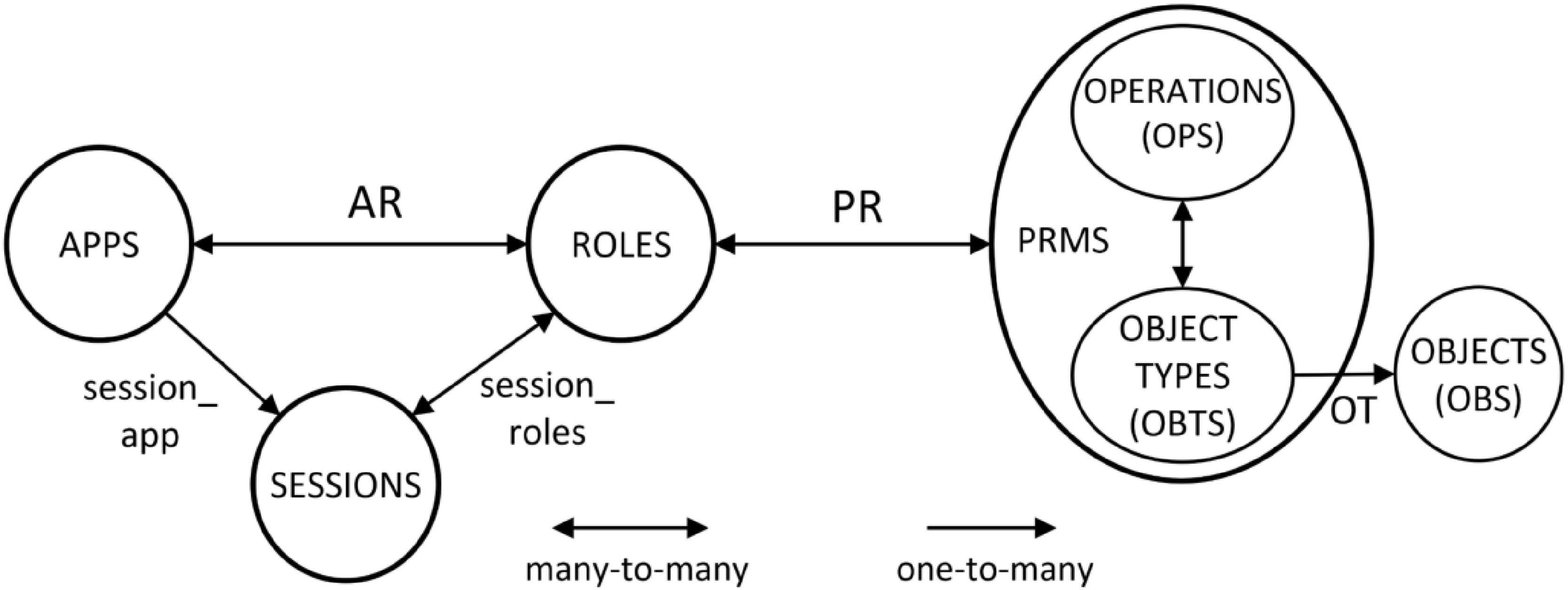


Total associations =3 x 6 = 18

Category	Permissions
Read	P_1 = read_topology P_2 = read_all_flow P_3 = read_statistics P_4 = read_pkt_in_payload
Notification	P_5 = pkt_in_event P_6 = flow_removed_event P_7 = error_event P_8 = topology_event
Write	P_9 = flow_mod_route P_{10} = flow_mod_drop P_{11} = flow_mod_modify_hdr P_{12} = modify_all_flows P_{13} = set_device_config P_{14} = set_flow_priority

Ahmad, Ijaz, Suneth Namal, Mika Ylianttila, and Andrei Gurtov. "Security in software defined networks: A survey." *IEEE Communications Surveys & Tutorials* 17, no. 4 (2015): 2317-2346.

SDN-RBAC Conceptual Model



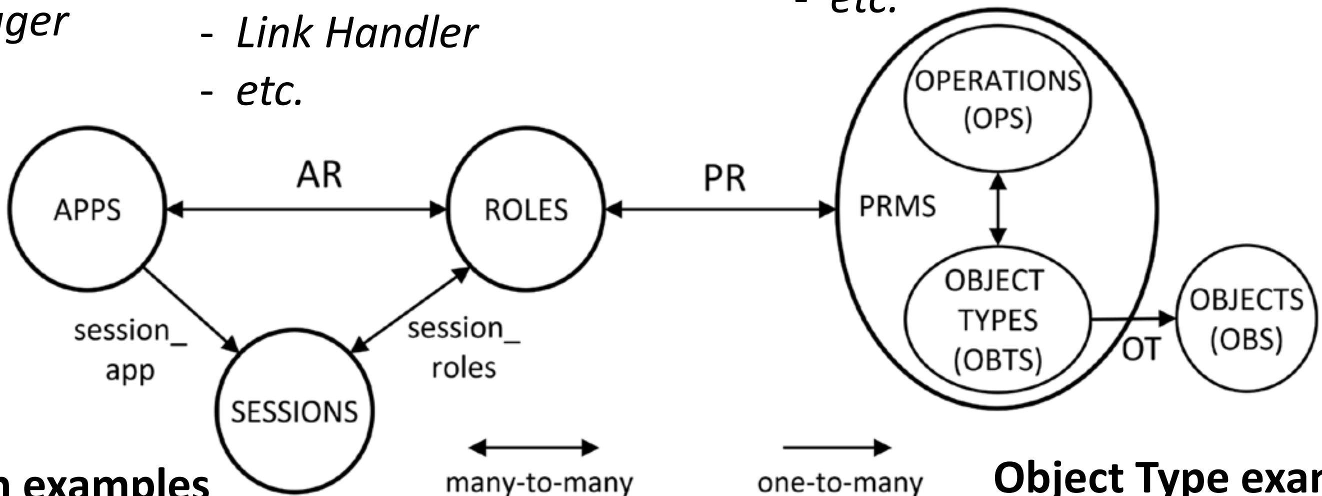
Al-Alaj, Abdullah, Ram Krishnan, and Ravi Sandhu. "SDN-RBAC: An Access Control Model for SDN Controller Applications." *2019 4th International Conference on Computing, Communications and Security (ICCCS)*. IEEE, 2019.

SDN-RBAC: Conceptual Model

App examples:
- *Routing app*
- *Load Balancing*
- *Topology Visualizer*
- *Network Debugger*
- *etc.*

Role examples:
- *Routing*
- *Flow Mod*
- *Device Handler*
- *Bandwidth Monitoring*
- *Link Handler*
- *etc.*

Operation examples:
- *Get Port BW Statistics*
- *Insert Flow to Switch*
- *get All Devices*
- *etc.*



Session examples
- *deep packet inspection session*
- *transmission rate monitoring session*
- *web-traffic filtering session*
- *shortest path precomputation session*
- *traffic redirection session*
- *etc*

Object Type example:
- *PORT*
- *FLOW-RULE*
- *LINK*
- *DEVICE*
- *PORT-STATS*
- *etc.*

Al-Alaj, Abdullah, Ram Krishnan, and Ravi Sandhu. "SDN-RBAC: An Access Control Model for SDN Controller Applications." *2019 4th International Conference on Computing, Communications and Security (ICCCS)*. IEEE, 2019.

SDN Apps and Sessions

- Users has no direct control of running controller apps.
- SDN apps usually perform several networking tasks.
- Apps could be compromised, buggy, or malicious.
- If executed in one session this means:
  - Higher exposure to the network attack surface.
- **Proposed solution:** cooperation of multiple sessions to perform all tasks of a complete SDN app.
  - Tasks can run sequentially or concurrently.
- **Goal**: minimize or avoid damage caused by one session.

RBAC Sessions for SDN apps: A motivation

The concept of a session has several motivations:

1. supports of least privilege principle:
   - delay the activation of roles currently unused in a session until they really required.
2. Delaying the creation of session itself until it is really required.

- **Goal:** Reduce the amount of permissions available to an app at a given time.
  - reduces the amount of resources accessible by these operations.
  - reduces exposure to network attack surface.

SDN-RBAC: Formal Definitions

Basic Element Sets | Assignment Relations | Mapping Functions

-	**Model Element Sets:**
-	$APPS$, $ROLES$, OPS, OBS and $OBTS$, a finite set of OpenFlow apps, roles, operations, objects and object types, respectively.
-	$PRMS = 2^{OPS \times OBTS}$, the set of permissions.
-	$SESSIONS$, a finite set of sessions.
-	**Assignment Relations:**
-	$PR \subseteq PRMS \times ROLES$, a many-to-many mapping permission-to-role assignment relation.
-	$AR \subseteq APPS \times ROLES$, a many-to-many mapping app-to-role assignment relation.
-	$OT \subset OBS \times OBTS$, a many-to-one relation mapping an object to its type.
-	**Mapping Functions**
-	$assigned\_perms(r : ROLES) \rightarrow 2^{PRMS}$, the mapping of role r into a set of permissions. Formally, $assigned\_perms(r) \subseteq \{p \in PRMS \mid (p, r) \in PR\}$.
-	$app\_sessions(a : APPS) \rightarrow 2^{SESSIONS}$, the mapping of an app into a set of sessions.
-	$session\_app(s : SESSIONS) \rightarrow APPS$, the mapping of session into the corresponding app.
-	$session\_roles(s : SESSIONS) \rightarrow 2^{ROLES}$, the mapping of session into a set of roles. Formally, $session\_roles(s) \subseteq \{r \in ROLES \mid (session\_app(s), r) \in AR\}$.
-	$type : OBS \rightarrow OBTS$, a function specifying the type of an object, where $(o, t_1) \in OT \wedge (o, t_2) \in OT \Rightarrow t_1 = t_2$
-	$\boxed{avail\_session\_perms(s : SESSIONS) \rightarrow 2^{PRMS}}$, the permissions available to an app in a session $= \bigcup_{r \in session\_roles(s)} assigned\_perms(r)$.

Very Important in Implementing checkAccess system function.

Al-Alaj, Abdullah, Ram Krishnan, and Ravi Sandhu. "SDN-RBAC: An Access Control Model for SDN Controller Applications." *2019 4th International Conference on Computing, Communications and Security (ICCCS)*. IEEE, 2019.

SDN-RBAC: Specifications of System Functions

- **Session Creation/Deletion**
- **Adding/Dropping Active Role**
- **Access Check**

Function	**Authorization Condition**	**Update**
$createSession(a : APPS, s : SESSIONS, ars : 2^{ROLES})$	$ars \subseteq \{r \in ROLES \mid (a, r) \in AR\} \wedge s \notin SESSIONS$	$SESSIONS' = SESSIONS \cup \{s\}, app\_sessions'(a) = app\_sessions(a) \cup \{s\}, session\_roles'(s) = ars$
$deleteSession(a : APPS, s : SESSIONS)$	$s \in app\_sessions(a)$	$app\_sessions'(a) = app\_sessions(a) \setminus \{s\}, SESSIONS' = SESSIONS \setminus \{s\}$
$addActiveRole(a : APPS, s : SESSIONS, r : ROLES)$	$s \in app\_tsessions(a) \wedge (a, r) \in AR \wedge r \notin session\_roles(s)$	$session\_roles'(s) = session\_roles(s) \cup \{r\}$
$dropActiveRole(a : APPS, s : SESSIONS, r : ROELS)$	$s \in app\_sessions(a) \wedge r \in session\_roles(s)$	$session\_roles'(s) = session\_roles(s) \setminus \{r\}$
$checkAccess(s : SESSIONS, op : OPS, ob : OBS)$	$\exists r \in ROLES : r \in session\_roles(s) \wedge ((op, \boxed{type(ob)}), r) \in PR$	

Apps are authorized based on object type.

Al-Alaj, Abdullah, Ram Krishnan, and Ravi Sandhu. "SDN-RBAC: An Access Control Model for SDN Controller Applications." *2019 4th International Conference on Computing, Communications and Security (ICCCS)*. IEEE, 2019.

Sessions in SDN-RBAC

- Two types:
  1. Atomic network sessions.
     - Self-contained task definition.
  2. Dependent network sessions.
     - Inter-session dependency.
     - Conduct inter-session interaction at runtime.

1. Atomic Sessions

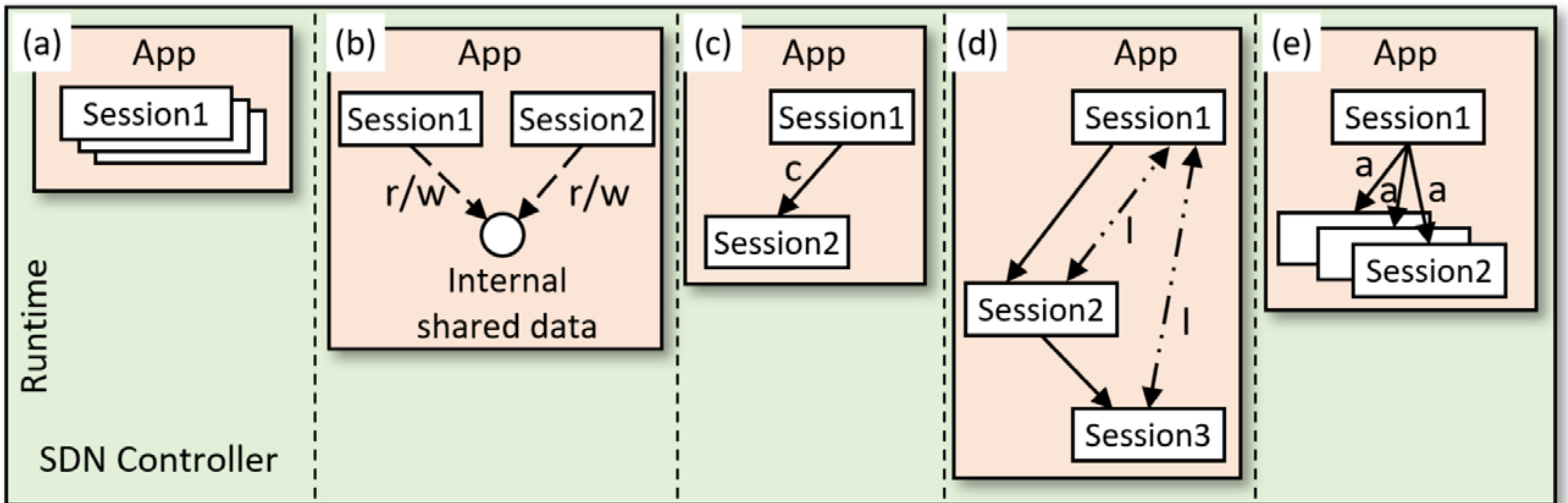
- Has self-contained task definition.
- Not dependent on other session instances.
  - a session that is not described in terms of other sessions
  - has no interaction with other session instances.
- Atomic sessions can have an **independent existence** and run **sequentially** or **simultaneously** without reference to each other.

2. Dependent Network Sessions

- Sessions has inter-session dependency.
- Conduct inter-session interaction at runtime.
- The execution of one session affects another one.
- Inter-session dependency initiates the need for:
  - methods for inter-session interaction.
  - conditions for session creation/deletion.
  - nomination of session’s active role set.
  - adding/dropping an active role.

Methods for Inter-session Interaction for SDN-RBAC

Atomic sessions	Two sessions access shared data	Conditional session creation	Interaction via inter-session interaction APIs	Active role set sent from master to slave sessions



⟶ : creates a session (From the creator to the created session).

– – – ▶ : access shared data.

◀ · · — ▶ : session interaction via session interaction API.

w/r : read/write operation.

c : condition that triggers session creation.

I : session interaction API (managed by the system).

a : active role set sent along with session creation request.

Al-Alaj, Abdullah, Ram Krishnan, and Ravi Sandhu. "SDN-RBAC: An Access Control Model for SDN Controller Applications." *2019 4th International Conference on Computing, Communications and Security (ICCCS)*. IEEE, 2019.

Session Handling Approaches

- Who is responsible of specifying:
  - **(T)** the **tasks** and corresponding sessions.
  - **(C)** the **condition** for session creation/deletion.
  - **(A)** the **active** role set.
  - **(R)** **role** to be added/dropped during execution.

Session Handling Approaches

Developer-driven Approach	System-driven Approach	Session-driven Approach
(T) DD	**(T)** CR	**(T)** DD
(C) DD	**(C)** CR	**(C)** SR
(A) DD	**(A)** CR	**(A)** SR
(R) DD	**(R)** CR	**(R)** SR

DD = determined by Developer at Design-time.
CR = determined by Controller at Run-time.
SR = determined by Session at Run-time.

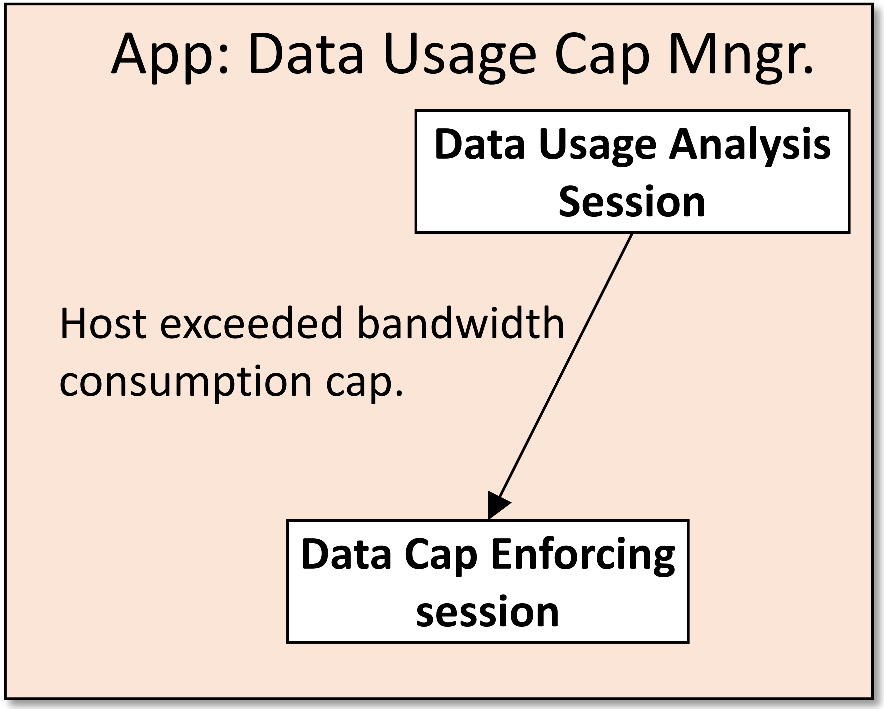
Developer-driven Session Handling (1)

The **developer** specifies at **design** time different session handling aspects:

(T) DD
(C) DD
(A) DD
(R) DD

- **[T]** task that will be achieved by each session.
  - Programmed by the **developer** at **design** time.
- **[C]** condition (or precondition) under which a particular session may be created/deleted
  - Condition is hard coded in the application.
  - Examples of session creation conditions include:
    - exceeding a bandwidth consumption cap,
    - new device detected,
    - at system start-up
- **[A]** active role set that should be activated during session creation
  - Example: ars = {"Packet-in Handler", "Flow Mod"} for a one-hour deep packet inspection session that will temporarily inspect traffic payload incoming from black-listed hosts.
- **[R]** adding or dropping an active role for a session
  - Example: add role "Device Hander" to a transmission rate monitoring session.

Developer-driven Session Handling (2) - Example



System-driven Session Handling (1)

(T) CR
(C) CR
(A) CR
(R) CR

- Controller has full control on session handling.
- The developer only provides the set of roles required by the app
- Developer has no discretion on determining any of other sessions' properties at runtime.
- Controller should be shipped with adequate capabilities to specify at runtime what session instances will be created and how to handle them.
- Given an app and the set of roles required by the app, the controller should figure out:
  - each task that might execute in a separate session
  - set of roles required for each session.
- This approach is challenging and the hardest to implement.

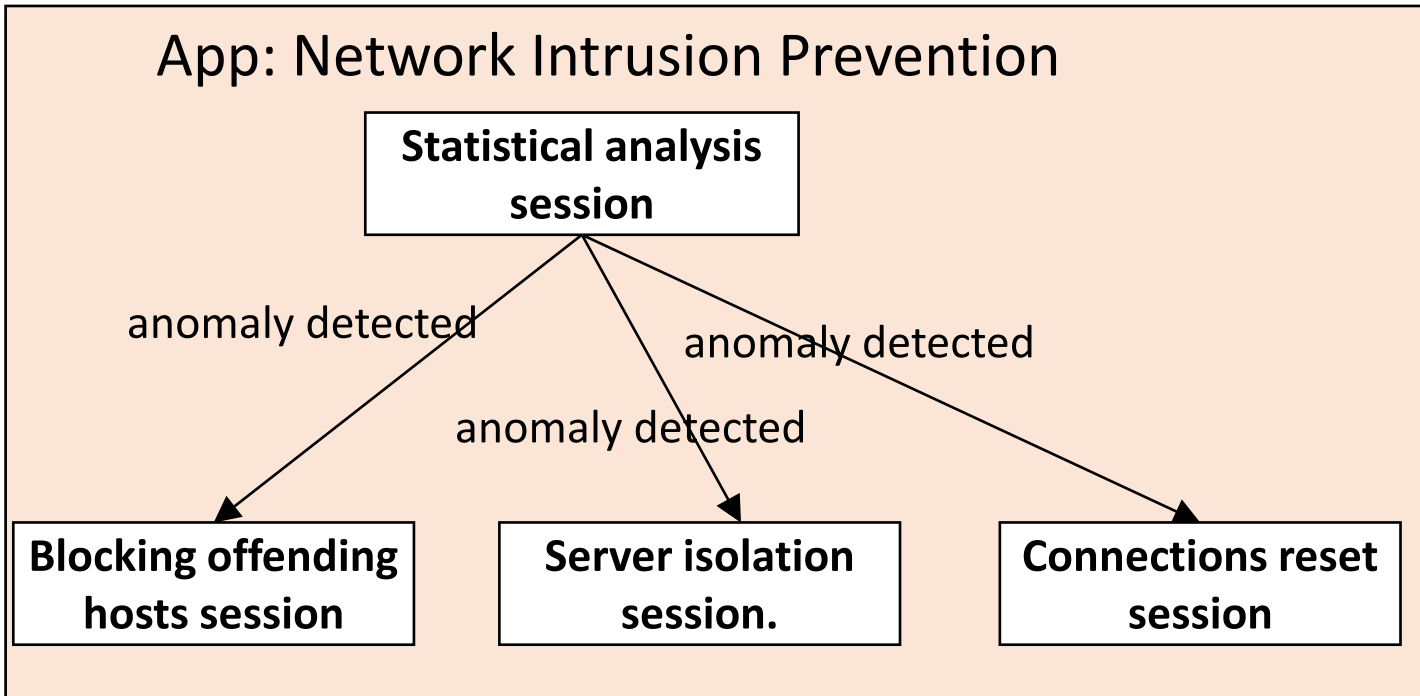
System-driven Session Handling (2)

The **controller** should specify dynamically at **runtime** the various properties:

(T) CR
(C) CR
(A) CR
(R) CR

- **[T]** The set of tasks required to achieve the entire app's functionality.
- **[C]** the condition under which a particular session instance may be created/deleted.
  - Examples : after attack detected, completion of another session, change of risk value, etc.
  - Developer doesn't know why a particular session could be created/terminated.
  - The criteria for creating a session could be computed dynamically by the controller or configured by the administrator at runtime.
- **[A]** active role set that should be activated during session creation.
  - Example: ars = {"Routing", "Link Handler"} for a session that *recomputes shortest path* after a new link discovery.
- **[R]** adding or dropping an active role for a session.

System-driven Session Handling (3)



Session-driven Approach (1)

- Inter-session interaction should be conducted via a well-defined set of session interaction APIs.
- The app **developer** should comply to these APIs during app **design**.
- Inter-session interaction APIs allow sessions to
  - Prob for information from other sessions, for example :
    - getting names of current active sessions.
    - getting active role set of a session.
    - getting session's status. (e.g., idle time, up time, etc).
  - Passing information and notifications between sessions. e.g., results of calculations.
- These sessions are smart.
  - take decisions based on the result of communications via inter-session interaction APIs.
  - adjust its behavior to take knowledgeable decisions on future session interaction API calls and on different session handling aspects.

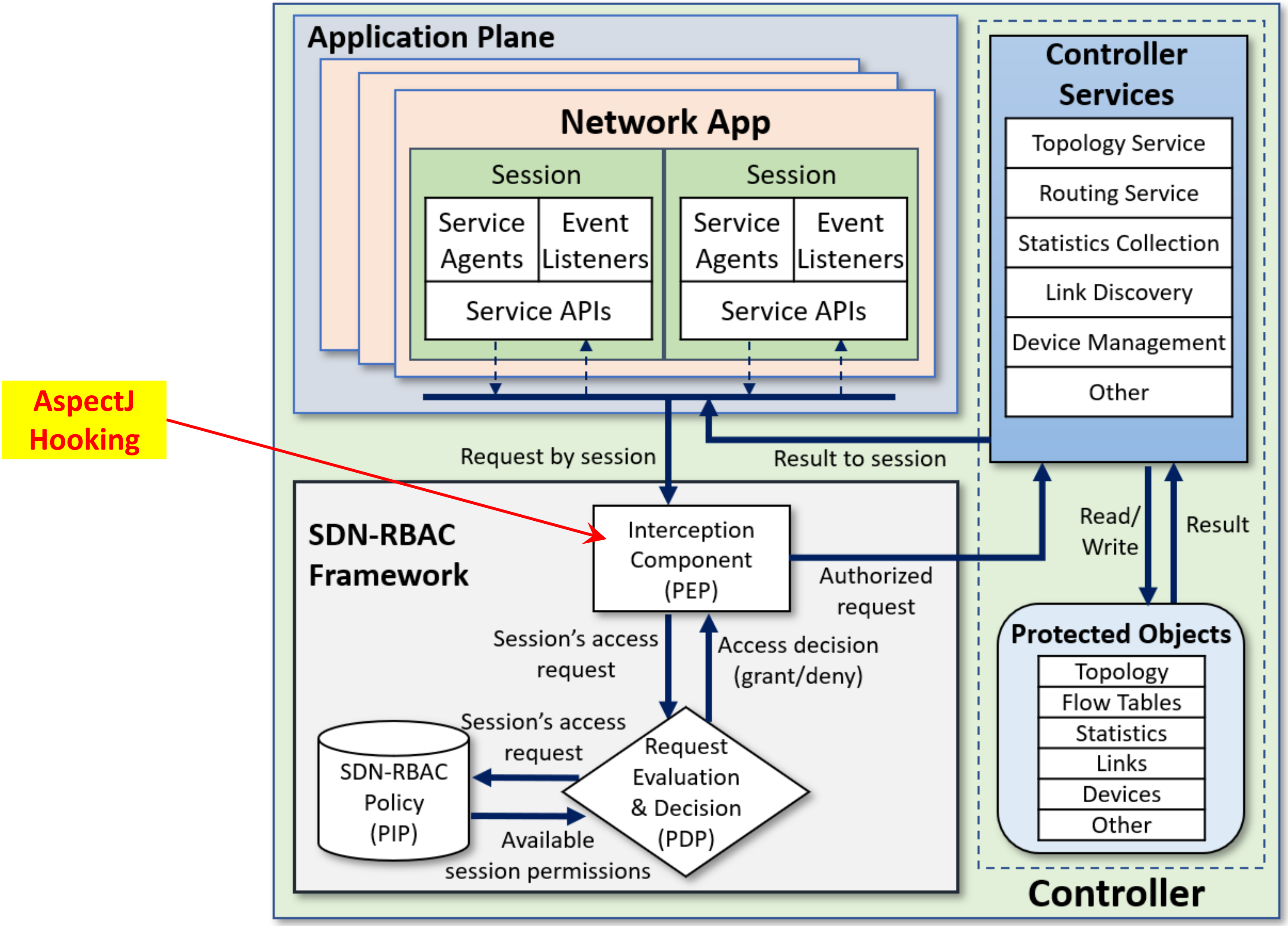
(T) DD
(C) SR
(A) SR
(R) SR

Session-driven Approach (2)

- Sessions adjust its behavior to take knowledgeable decisions on future session interaction API calls and on the following session handling aspects:
- **[T]** task that will be achieved by each session.
  - Programmed by the **developer** at **design** time.
  - The app **developer** should comply to session interaction APIs during app **design**.
- **[C]** condition under which a particular session instance may be created/deleted
  - Example: start *traffic redirection session* after an alarm is fired by *packet inspection session*.
- **[A]** active role set that should be activated during session creation
  - Example: ars = {“Packet-in Handler”, “Flow Mod”} role for a *deep packet inspection session* if *web-traffic filtering session* detected malicious payloads.
- **[R]** adding or dropping an active role for a session.

(T) DD
(C) SR
(A) SR
(R) SR

SDN-RBAC Framework Architecture



Al-Alaj, Abdullah, Ram Krishnan, and Ravi Sandhu. "SDN-RBAC: An Access Control Model for SDN Controller Applications." *2019 4th International Conference on Computing, Communications and Security (ICCCS)*. IEEE, 2019.

SDN-RBAC Model Characteristics

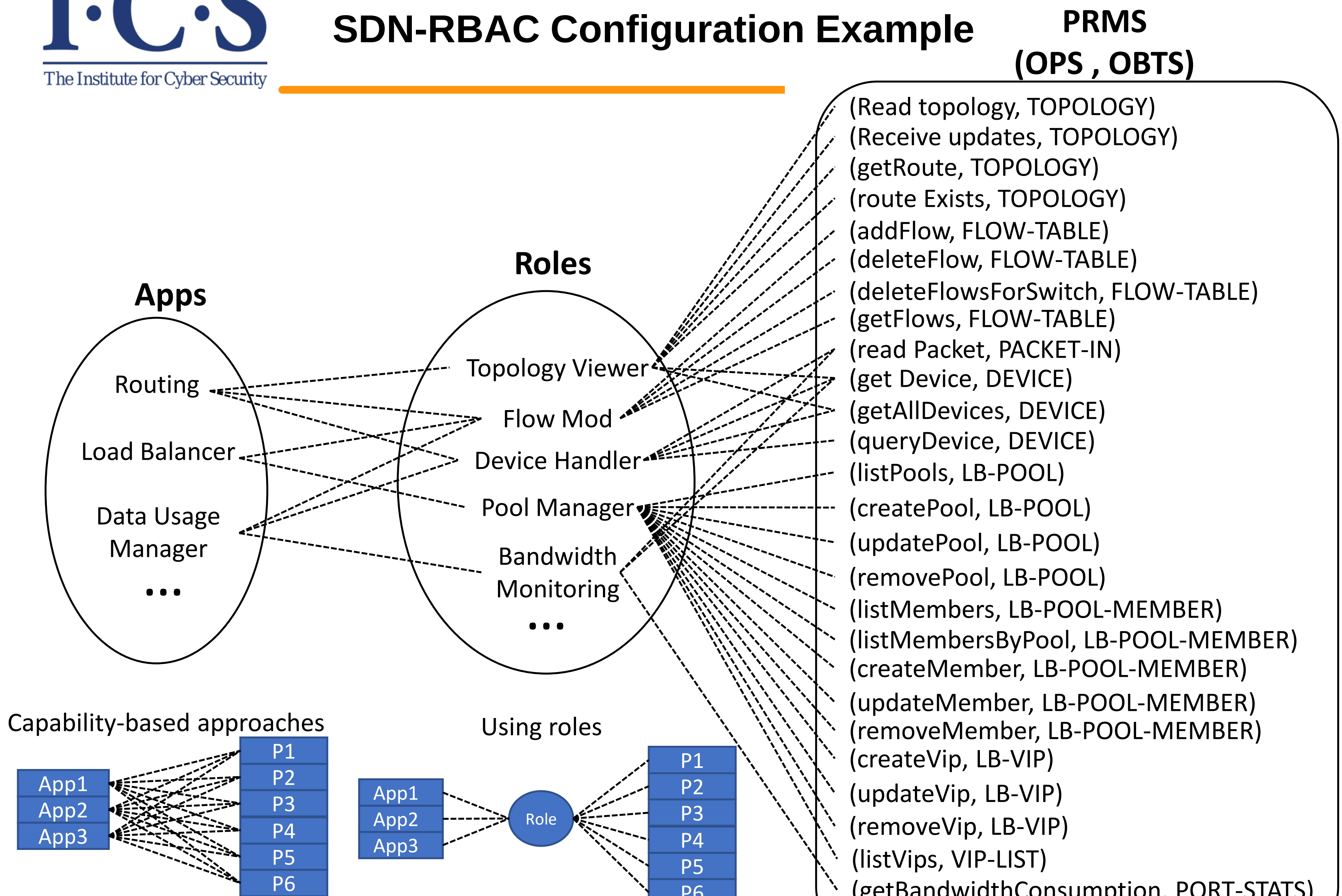
- **SDN-RBAC model and implementation promotes SDN authorization system by**:
  - Covering northbound and southbound interfaces.
  - Defining sufficient roles.
  - Supporting Many-to-Many app-to-role and permission-to-role relations.
  - The use of sessions.
  - The use of expressive roles compliant with various network functions – helps in creating expressive security policy.

Selected roles in SDN-RBAC

Role	General Functionality
Device Handler	permissions for querying the controller about devices
Bandwidth Monitoring	permissions to read the bandwidth consumption for switch ports.
Flow Mod	permissions to insert/update/delete flow rules into a switch's flow tables.
Link Handler	permissions to get information about network links
Device Tracking	permissions to get notifications about changes on network devices (added, removed, Moved, Address Changed, etc.)
Port Handler	permissions to read information about ports and their status
Routing	permissions to get and compute routes between various source and destination nodes

Al-Alaj, Abdullah, Ram Krishnan, and Ravi Sandhu. "SDN-RBAC: An Access Control Model for SDN Controller Applications." *2019 4th International Conference on Computing, Communications and Security (ICCCS)*. IEEE, 2019.

SDN-RBAC Configuration Example

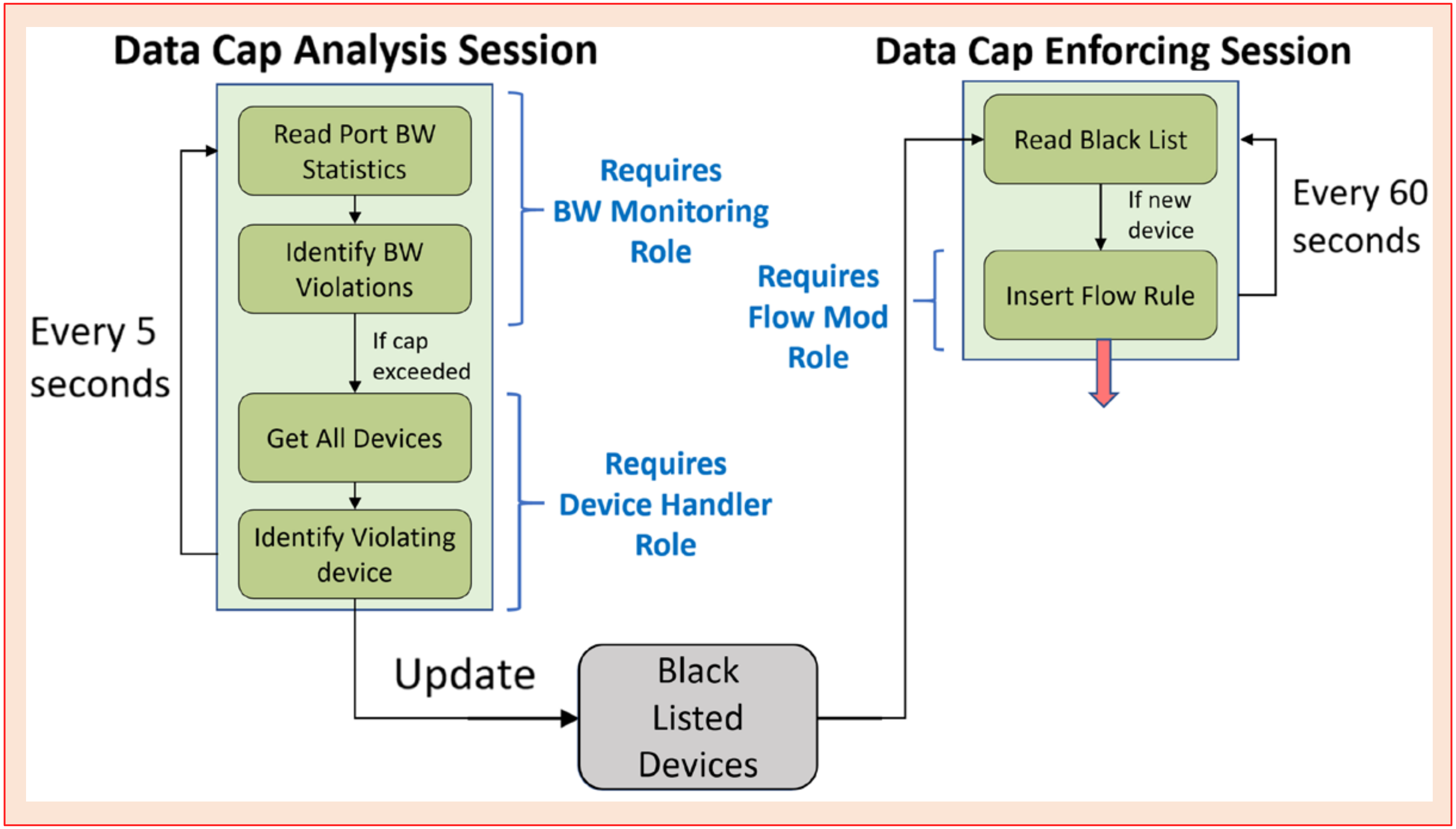


Total associations =3 x 6 = 18

Total associations = 3 + 6 = 9

Access control use case

Data Usage Cap Manager - Multi-session App



Data Usage Cap Manager Use-case app

- App developed in Floodlight: DataUsageCapMngr.
- **Session1**: DataUsageAnalysisSession:
  - probes for port bandwidth statistics on a regular basis (every 5 seconds). After reading the bandwidth consumption for switch ports and analyzing the data to find cap limit violations, it stores the list of hosts who has exceeded the cap limit into a list ‘usageCapBlackList’ managed by the system.
  - Required roles: “*Device Handler”* and “*Bandwidth Monitoring”.*
- **Session2**: DataCapEnforcingSession:
  - checks periodically (every 60 seconds) for black-listed hosts stored in ‘usageCapBlackList’.
  - inserts flow rules to isolate violating hosts from the network.
  - Required roles: “*Flow Mod*”.
- SDN-RBAC authorization system:
  - mediates all sessions access requests.
  - identifies each session,
  - sends sessions request for authorization check.
  - allows or blocks the request based on PDP response.

Use Case: Configuration in SDN-RBAC

Center for Security and Privacy

- **Use case sets:**
- $APPS = \{DataUsageCapMngr\}$.
- $ROLES = \{Device\ Handler, Bandwidth\ Monitoring, Flow\ Mod\}$.
  D = set of all network devices. FT = set of all flow tables in all switches, PS = set of all port statistics in all switches.
- $OBS = \{D, FT, PS\}$.
- $OBTS = \{DEVICE, PORT\text{-}STATS, FLOW\text{-}TABLE\}$.
- $OT = \{(D, DEVICE), (PS, PORT\text{-}STATS), (FT, FLOW\text{-}TABLE)\}$.
- **Permissions:**
- $PRMS = \{p_1, p_2, p_3\}^1$ with
  $p_1 = (getAllDevices, DEVICE), p_2 = (getBandwidthConsumption, PORT\text{-}STATS), p_3 = (InsertRule, FLOW\text{-}TABLE)$.
- **Permissions assignment:**
- $PR = \{(p_1, Device\ Handler), (p_2, Bandwidth\ Monitoring), (p_3, Flow\ Mod)\}$.
- $assigned\_perms(Device\ Handler) = \{p_1\}^1, assigned\_perms(Bandwidth\ Monitoring) = \{p_2\}^1, assigned\_perms(Flow\ Mod) = \{p_3\}^1$
- **Role assignment:**
- $AR = \{(DataUsageCapMngr, Device\ Handler)$
  $(DataUsageCapMngr, Bandwidth\ Monitoring), (DataUsageCapMngr, Flow\ Mod)\}$. → **3 roles**
- **Sessions:**
- $SESSIONS = \{DataUsageAnalysisSession, DataCapEnforcingSession\}$. → **2 sessions**
- $app\_sessions(DataUsageCapMngr) = \{DataUsageAnalysisSession, DataCapEnforcingSession\}$. → **2 sessions**
- $session\_app(DataUsageAnalysisSession) = \{DataUsageCapMngr\}$,
  $session\_app(DataCapEnforcingSession) = \{DataUsageCapMngr\}$.
- **Active role sets:**
- $session\_roles(DataUsageAnalysisSession) = \{Device\ Handler, Bandwidth\ Monitoring\}$. → **2 roles**
- $session\_roles(DataCapEnforcingSession) = \{Flow\ Mod\}$. → **1 role**

[1] Sets with this mark in the table include minimum elements enough to understand the use case. Remaining elements are avoided for more convenience and readability.

THE CONFIGURATION OF THE $DataUsageCapMngr$ AND ITS TWO SESSIONS AS A USE CASE IN SDN-RBAC[1].

UTSA Computer Science

Al-Alaj, Abdullah, Ram Krishnan, and Ravi Sandhu. "SDN-RBAC: An Access Control Model for SDN Controller Applications." *2019 4th International Conference on Computing, Communications and Security (ICCCS)*. IEEE, 2019.

Usability demonstration (1) Access Denies

- To show that SDN-RBAC authorization system can identify and reject any unauthorized operations:
- We forced DataUsageAnalysisSession to read link information via operation **getAllLinks**.
- The permission (**getAllLinks**, LINK) is assigned to the role **LinkHandler**.
- Role **LinkHandler** is not a member of the active role set of DataUsageAnalysisSession.
- A snapshot of the execution result is shown below.

Unauthorized

Snapshot1

```
The method net.floodlightcontroller.topology.ITopologyService.getAllLinks
is called by session net.floodlightcontroller.datausagemngr.DataUsageAnalysisSession
16:36:31.982 WARN [n.f.rbac.RBAC:Thread-12] SDN-RBAC: security violation, "Access denied".
Unauthorized access requested by session (DataUsageAnalysisSession)
Reason: None of session active roles contains a corrseponding permission
Active roles set for this session: [Device Handler, Bandwidth Monitoring]
```

Snapshot of authorization check result for *getAllLinks()* operation requested by $DataUsageAnalysisSession$ - Access Denied.

Usability demonstration (2) Access Allowed

I·C·S The Institute for Cyber Security

C·SPECC Center for Security and Privacy Enhanced Cloud Computing

- We forced DataUsageAnalysisSession to read device statistics via operation **getBandwidthConsumption**.
- The permission (**getBandwidthConsumption**, PORT-STATS) is assigned to the role **BandwidthMonitoring**.
- Role **BandwidthMonitoring** is a member of the active role set of DataUsageAnalysisSession.
- A snapshot of the execution result is shown below.
- The snapshot below shows how DataUsageAnalysisSession was able to pass the authorization.

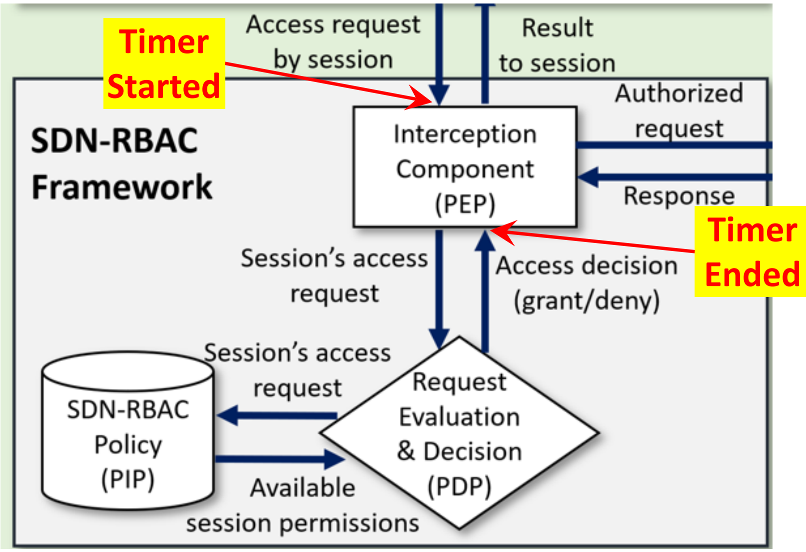
Authorized

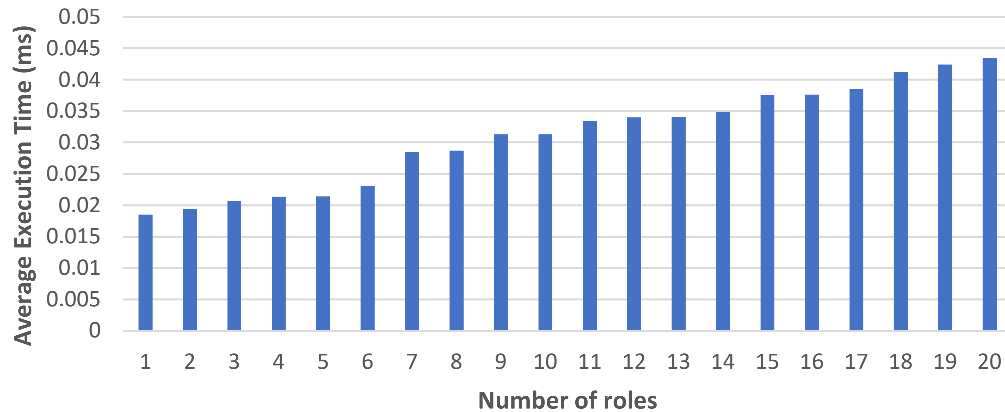
Snapshot2

```
The method net.floodlightcontroller.statistics.IStatisticsService.getBandwidthConsumption
is called by session net.floodlightcontroller.datausagemngr.DataUsageAnalysisSession
16:36:25.979 INFO [n.f.rbac.RBAC:Thread-12] SDN-RBAC: "Access granted": Authorized access
requested by session (DataUsageAnalysisSession)
Reason: The session role [Bandwidth Monitoring] contains the permission (net.floodlightcon
troller.statistics.IStatisticsService.getBandwidthConsumption, PORT-STATS)
```

Snapshot of authorization check result for *getBandwidthConsumption()* operation requested by *DataUsageAnalysisSession* - Access Granted.

SDN-RBAC Average Authorization Time





Average execution time required by SDN-RBAC components to check for authorization of 50 operations with varying number of roles.

On average: 0.031 ms overhead for 50 operations.

Al-Alaj, Abdullah, Ram Krishnan, and Ravi Sandhu. "SDN-RBAC: An Access Control Model for SDN Controller Applications." *2019 4th International Conference on Computing, Communications and Security (ICCCS)*. IEEE, 2019.

References

- Al-Alaj, Abdullah, Ram Krishnan, and Ravi Sandhu. "SDN-RBAC: An Access Control Model for SDN Controller Applications." *2019 4th International Conference on Computing, Communications and Security (ICCCS)*. IEEE, 2019.