SDN-RBAC: An Access Control Model for SDN Controller Applications

Abdullah Al-Alaj[1], Ram Krishnan[2] and Ravi Sandhu[1]

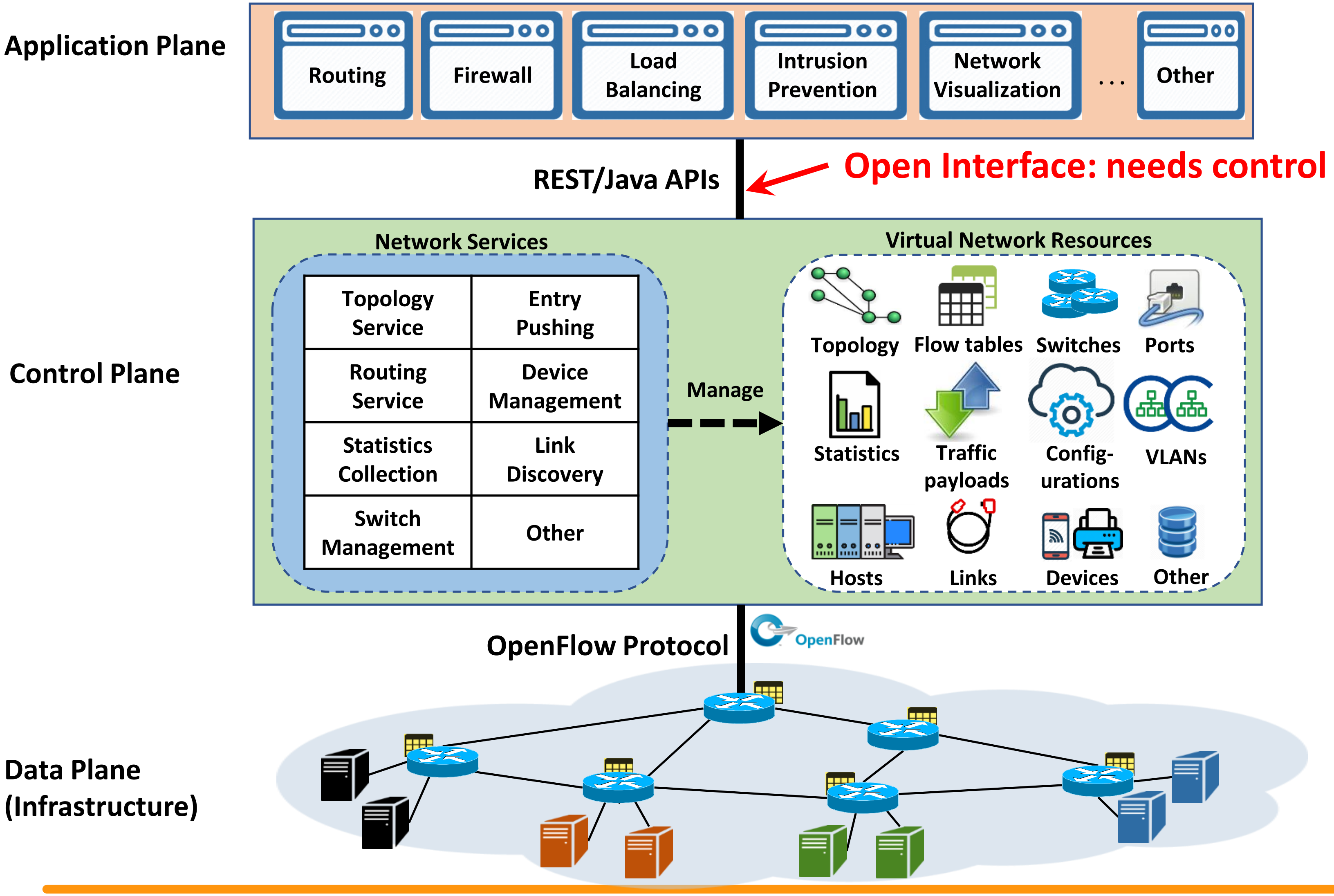
[1]Dept. of Computer Science
[2]Dept. of Electrical and Computer Engineering
[1,2]Institute for Cyber Security
[1,2]Center for Security and Privacy Enhanced Cloud Computing (C-SPECC)
University of Texas at San Antonio, TX 78249

4th IEEE International Conference on Computing, Communication and Security (ICCCS'2019)
Rome, Italy, October 10-12, 2019

Agenda

- Introduction
- Access Control for SDN
- SDN-RBAC Model
- App Sessions in SDN-RBAC
- SDN-RBAC System Architecture
- Use Case and Configuration
- Performance Evaluation
- Conclusion and Future Work

Introduction



Application Plane

REST/Java APIs

Open Interface: needs control

Control Plane

Network Services

Topology Service	Entry Pushing
Routing Service	Device Management
Statistics Collection	Link Discovery
Switch Management	Other

Manage

Virtual Network Resources

OpenFlow Protocol

Data Plane (Infrastructure)

Access control for SDN

- **Access control problem:**
  - Control which subjects (SDN apps) can access which objects (virtual network resources) for performing which actions (SDN operations).
- **Key issues for SDN include:**
  - Reducing network exposure to attack surface.
    - Apply principle of least privileges for SDN apps.
    - Minimize active permissions available for an SDN app.
  - Facilitate administration of access control.
- **Challenges:**
  - Handling sessions of controller apps (no direct user interaction).
  - Implementing access control with minimal change to controller’s code.

SDN-RBAC: Conceptual Model

App examples:

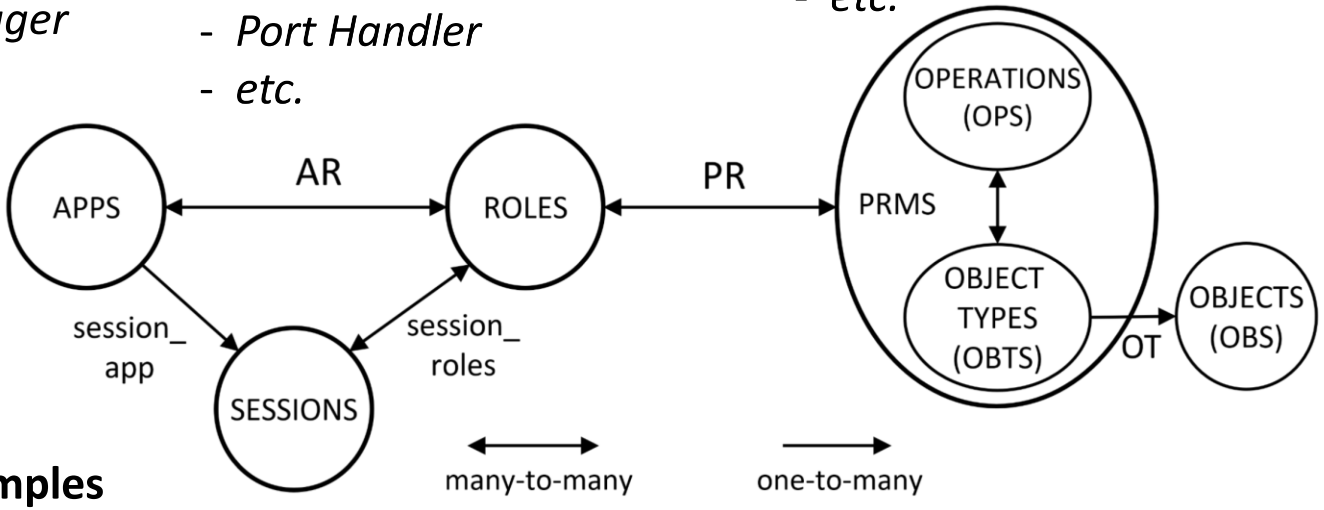
- *Routing app*
- *Load Balancing*
- *Topology Visualizer*
- *Network Debugger*
- *etc.*

Role examples:

- *Routing*
- *Device Handler*
- *Bandwidth Monitoring*
- *Link Handler*
- *Port Handler*
- *etc.*

Operation examples:

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



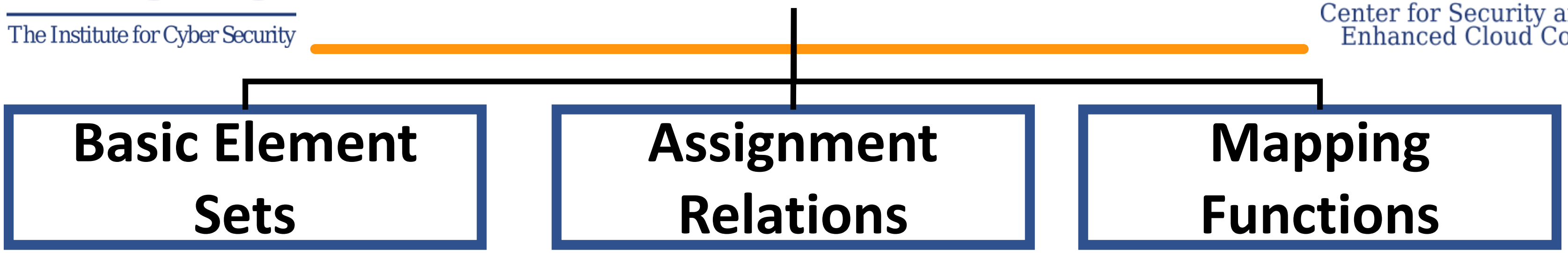
Session examples

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

Object Type example:

- *PORT- VLAN-5, PORT-VLAN-10*
- *LINK-CS, LINK-ACC*
- *HOST-TENANT-X, HOST-TENANT-Y*
- *etc.*

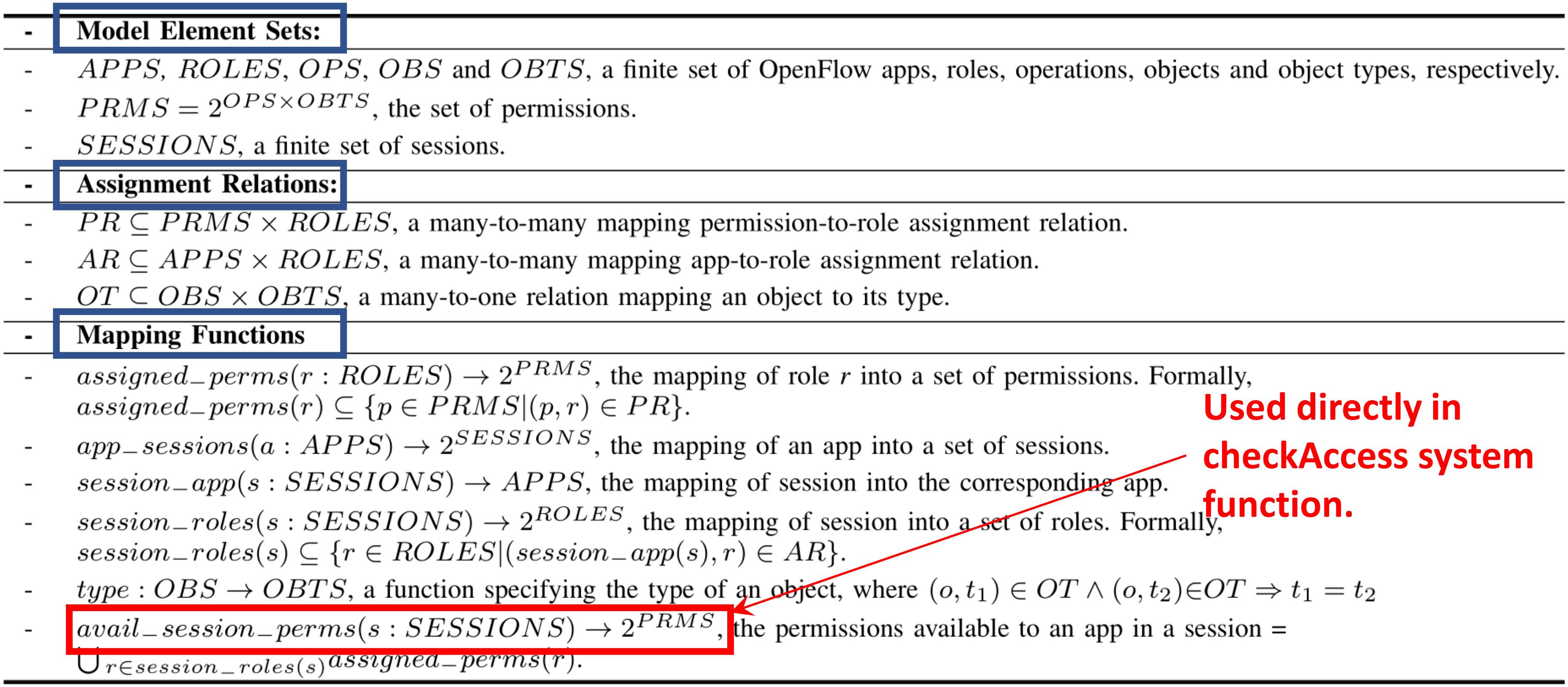
SDN-RBAC: Formal Definitions



Basic Element Sets

Assignment Relations

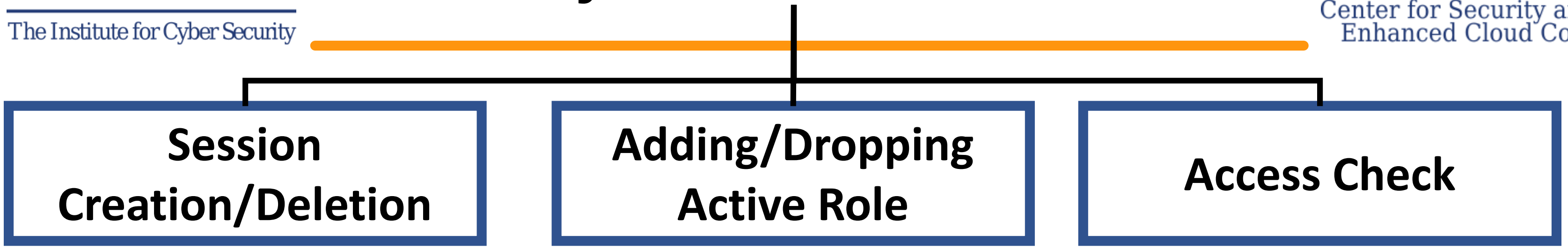
Mapping Functions



- **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, $assigned\_perms(r) \subseteq \{p \in PRMS | (p, r) \in PR\}$.
- $app\_sessions(a : APPS) \rightarrow 2^{SESSIONS}$, 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, $session\_roles(s) \subseteq \{r \in ROLES | (session\_app(s), r) \in AR\}$.
- $type : OBS \rightarrow OBTS$, a function specifying the type of an object, where $(o, t_1) \in OT \wedge (o, t_2) \in OT \Rightarrow t_1 = t_2$
- $avail\_session\_perms(s : SESSIONS) \rightarrow 2^{PRMS}$, the permissions available to an app in a session = $\bigcup_{r \in session\_roles(s)} assigned\_perms(r)$.

Used directly in checkAccess system function.

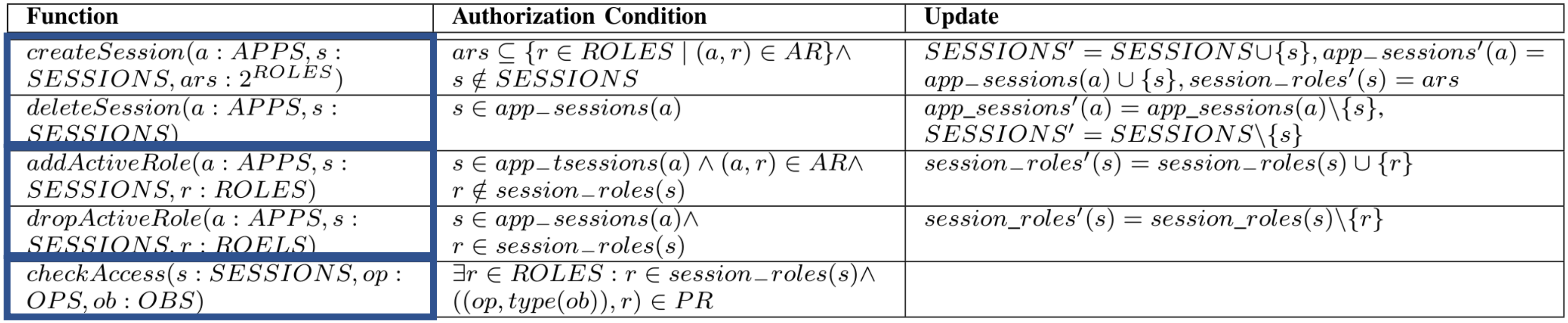
SDN-RBAC: Specifications of System Functions



Session Creation/Deletion

Adding/Dropping Active Role

Access Check



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

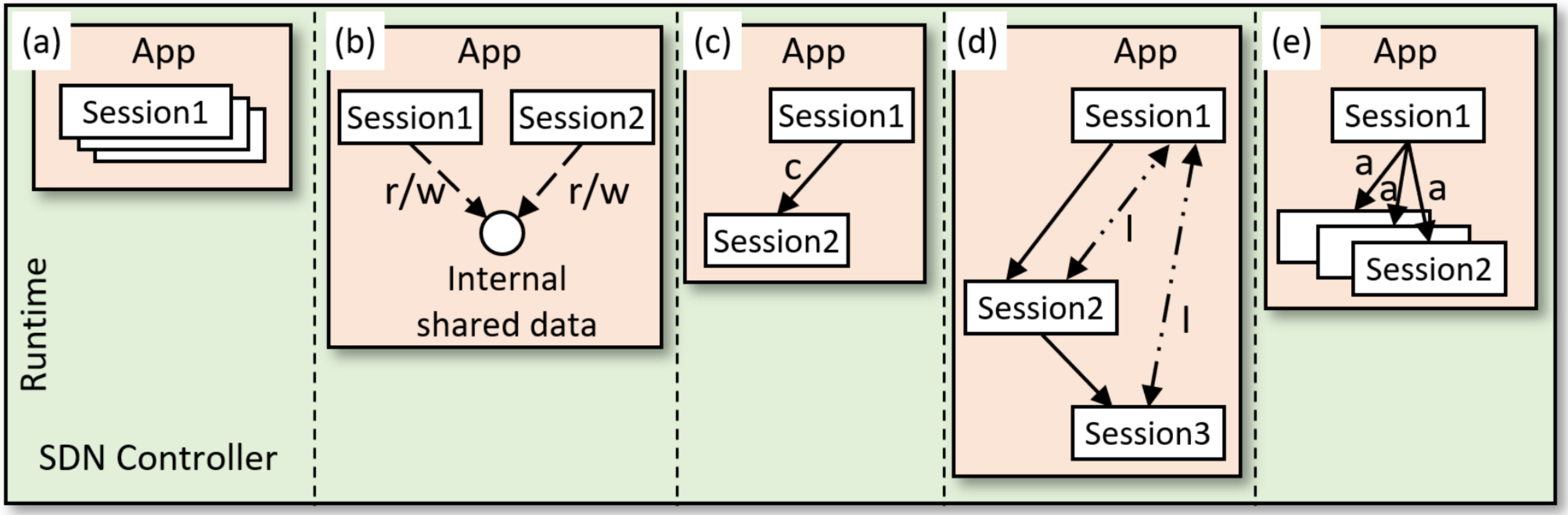
retrieving the object type

Sessions in SDN-RBAC

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

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.

Methods for Inter-session Interaction for SDN-RBAC

Examples of conditions for session creation:

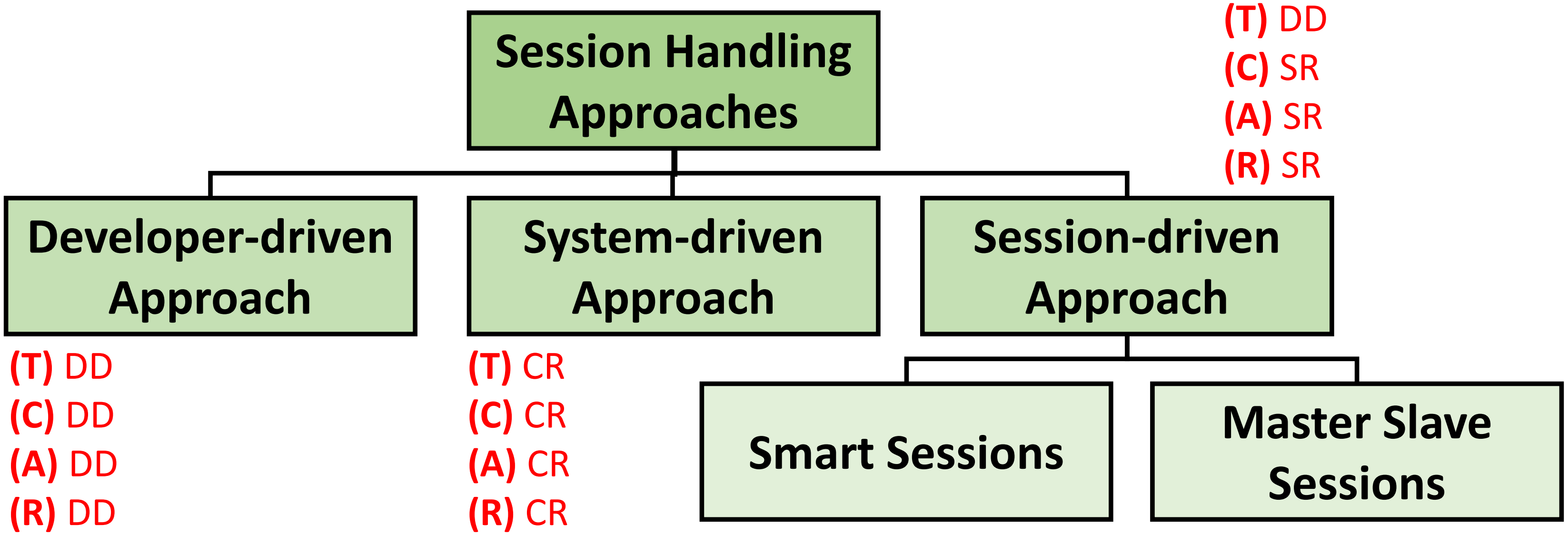
- bandwidth consumption cap exceeded,
- new device detected,
- at system start-up.
- etc.

Session handling APIs usage examples:

- Getting names of all 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.

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

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

System-driven Approach

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

Session-driven Approach

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

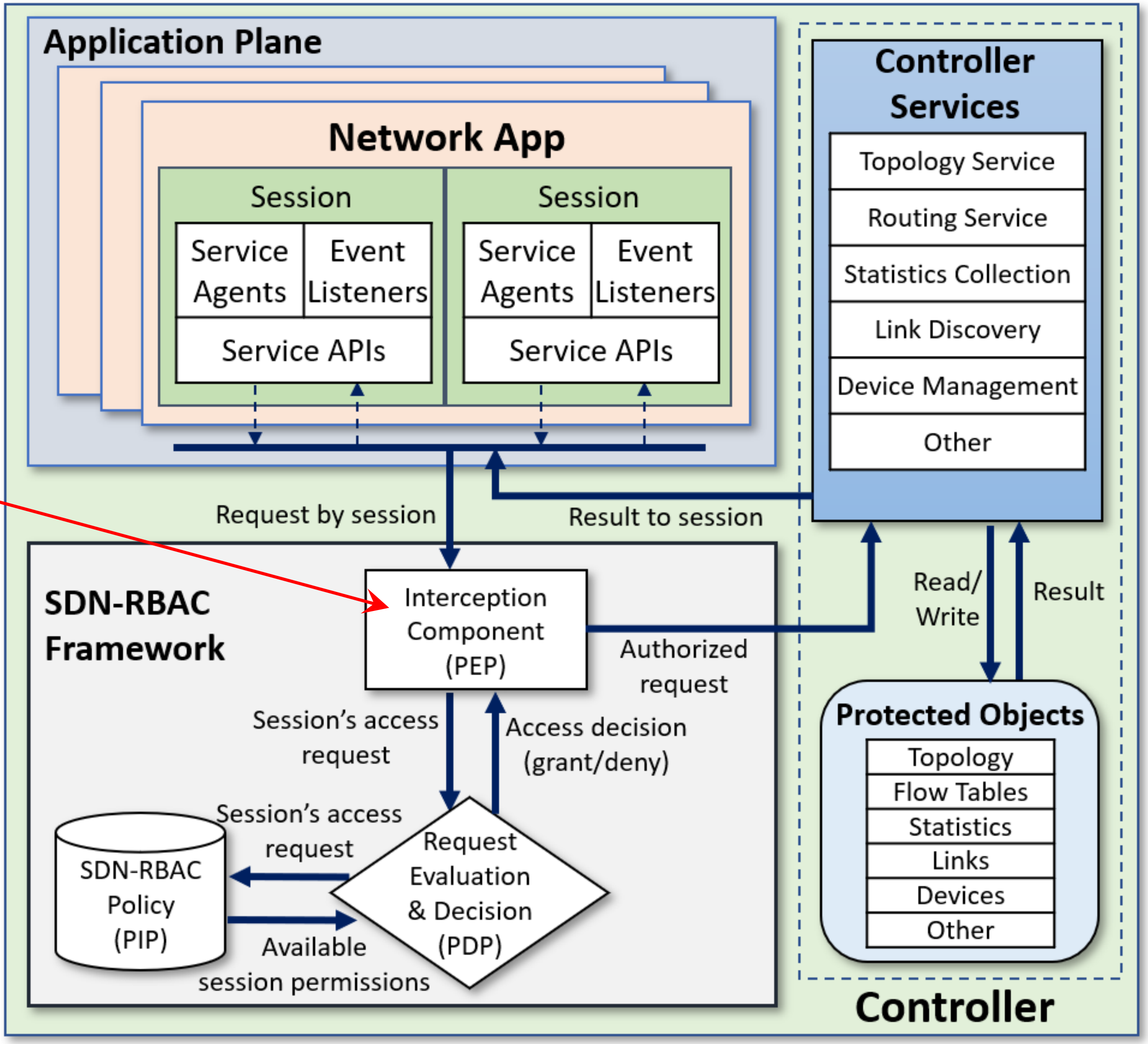
Smart Sessions

Master Slave Sessions

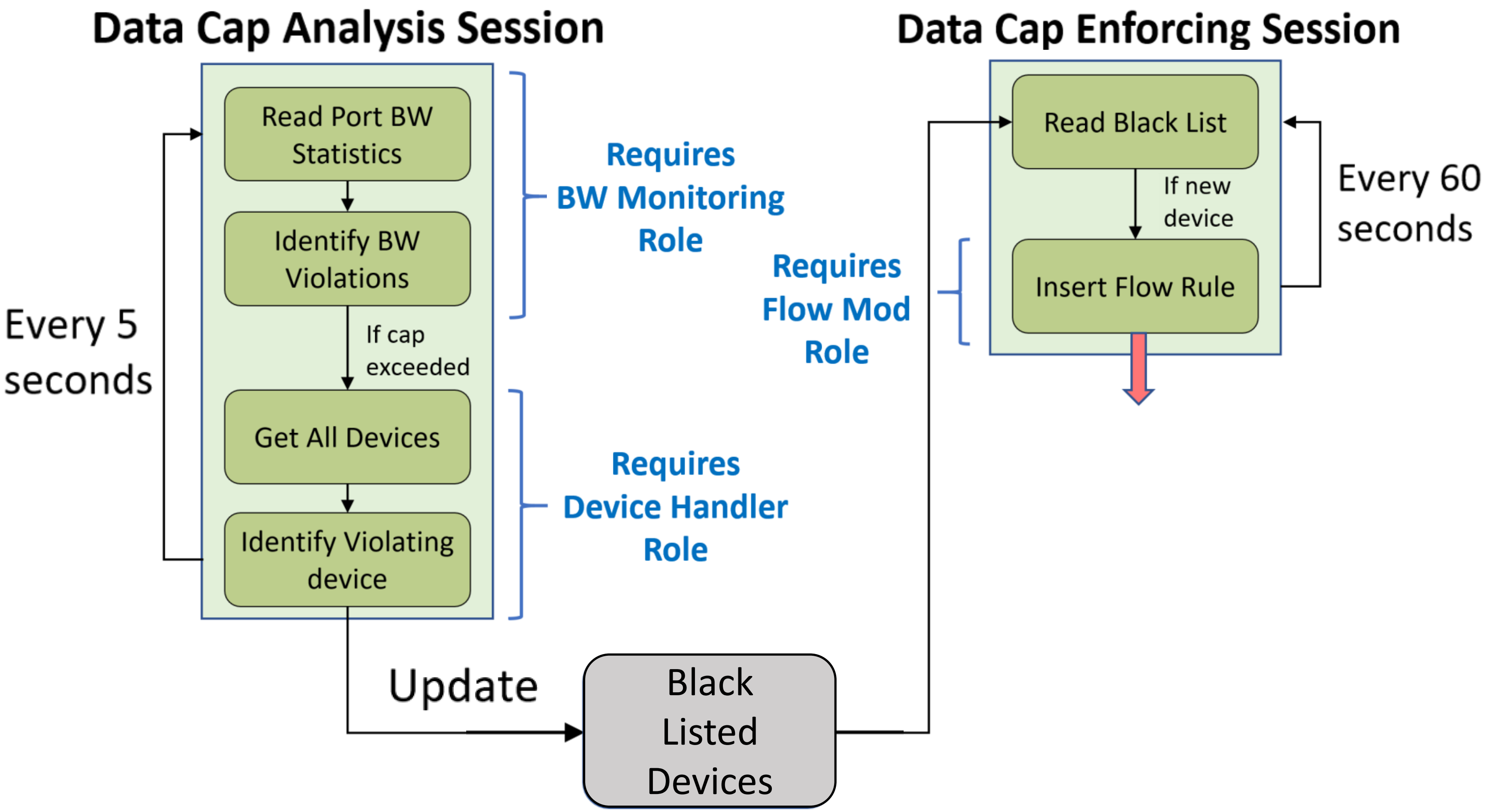
DD – determined by developer at design-time.
CR – determined by controller at run-time.
SR – determined by session at run-time.

SDN-RBAC: System Architecture

AspectJ Hooking



Use Case: Data Usage Manager (A Multi-session App)



Data Cap Analysis Session

Data Cap Enforcing Session

Requires BW Monitoring Role

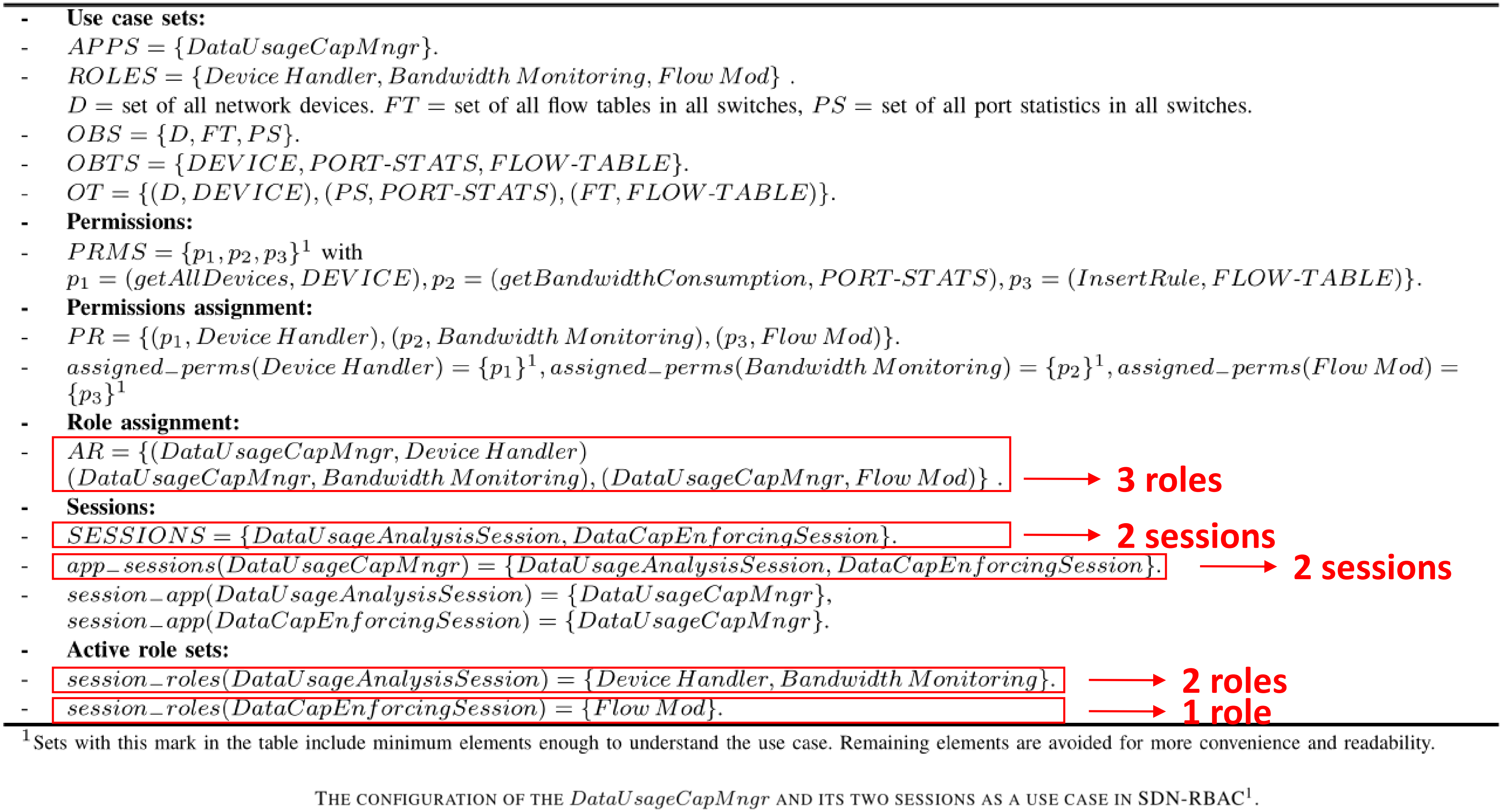
Requires Device Handler Role

Requires Flow Mod Role

Every 5 seconds

Every 60 seconds

Use Case: Configuration in SDN-RBAC



- **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

- $session\_roles(DataCapEnforcingSession) = \{Flow\ Mod\}$.

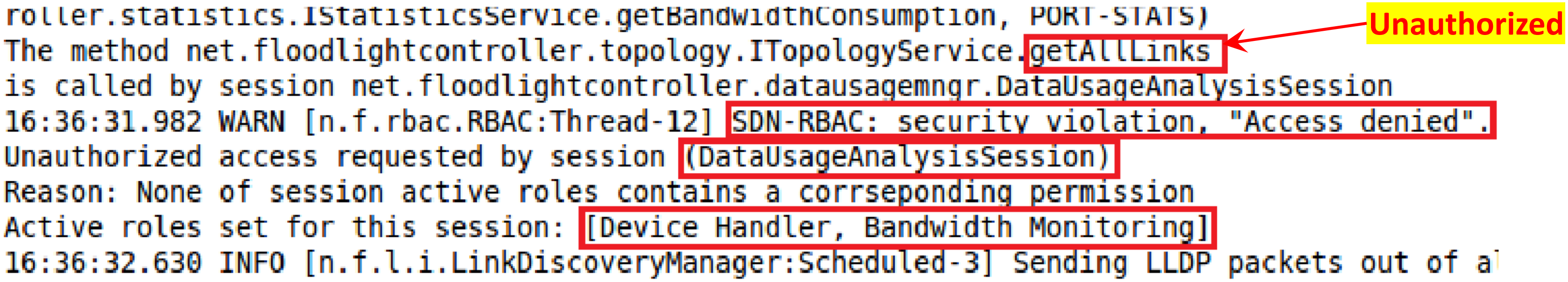
1 role

[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].

Demonstration in Floodlight: Data Cap Manager App

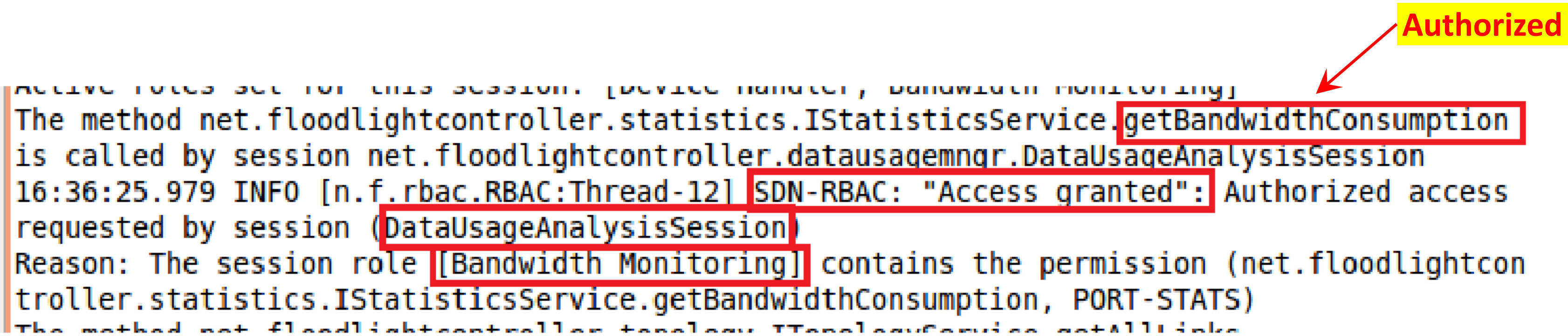
Snapshot1



```
roller.statistics.IStatisticsService.getBandwidthConsumption, PORT-STATS)
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]
16:36:32.630 INFO [n.f.l.i.LinkDiscoveryManager:Scheduled-3] Sending LLDP packets out of a
```

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

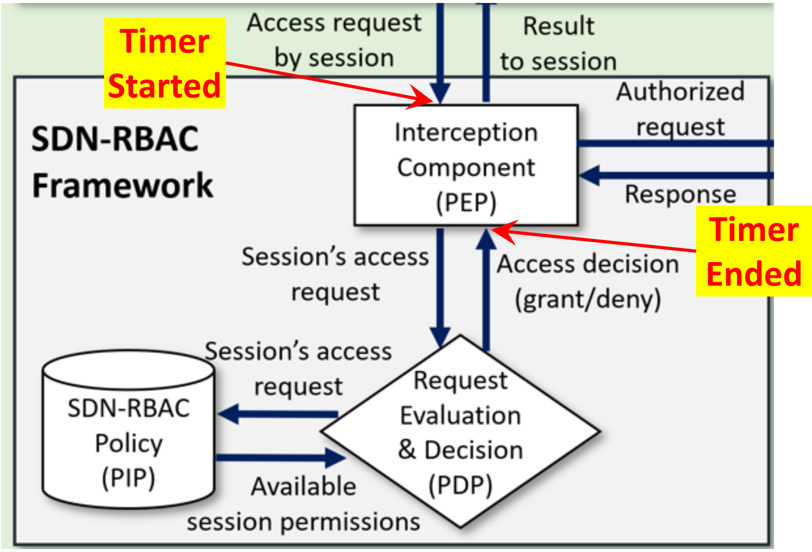
Snapshot2

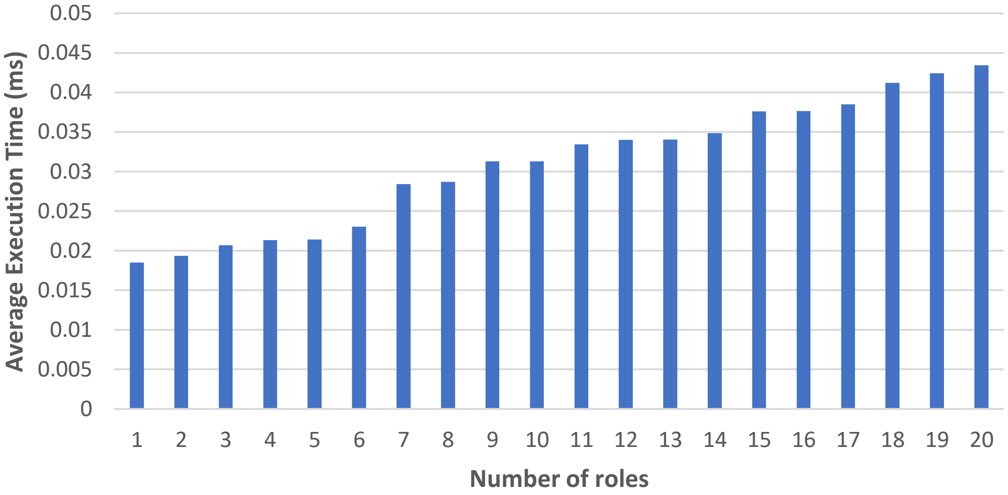


```
Active roles set for this session: [Device Handler, Bandwidth Monitoring]
The method net.floodlightcontroller.statistics.IStatisticsService.getBandwidthConsumption
is called by session net.floodlightcontroller.datausagemngr.DataUsageAnalysisSession
16:36:25.979 INFO [n.f.rbac.RBAC:Thread-12] SDN-RBAC: "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)
The method net.floodlightcontroller.topology.ITopologyService.getAllLinks
```

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

SDN-RBAC Average Decision Time

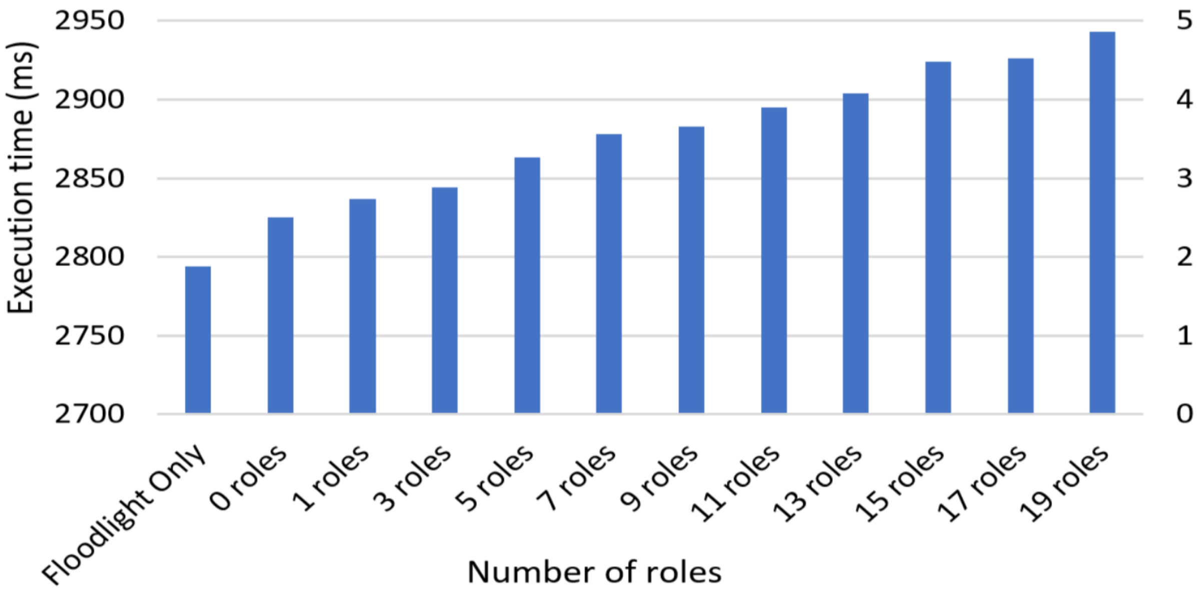




Average execution time required by SDN-RBAC components to finish checking 50 operations with varying number of roles.

On average: 0.031 ms overhead for 50 operations.

Controller with SDN-RBAC Performance Evaluation



Average total execution time required to finish the 50 operations called 1000 times including and excluding SDN-RBAC.

Conclusion and Future Work

In this work:

- A formal model (SDN-RBAC) for SDN controller apps.
- Methods for Inter-session Interaction.
- Different approaches for handling session instances of an app.
- Implementation of the model, as proof-of-concept prototype, in Floodlight platform.
- We used hooking techniques without any change to the code of Floodlight native modules.
- We show the system’s usability using a test app with multi-session execution.
- Performance evaluation with various number of roles.

Future research

- Hierarchical priority groups for conflict resolution between apps operations.
- Role-based administration of SDN-RBAC and its extensions.