Role Based Access Control For Software Defined Networking Formal Models and Implementation

Dissertation Defense
Abdullah Al-Alaj
Institute for Cyber Security
Department of Computer Science
The University of Texas at San Antonio

Committee:
Prof. Ravi Sandhu, Ph.D. (Advisor)
Dr. Ram Krishnan, Ph.D. (Co-advisor)
Dr. Palden Lama, Ph.D.
Prof. Gregory White, Ph.D.
Dr. Weining Zhang, Ph.D.

July 20, 2020.

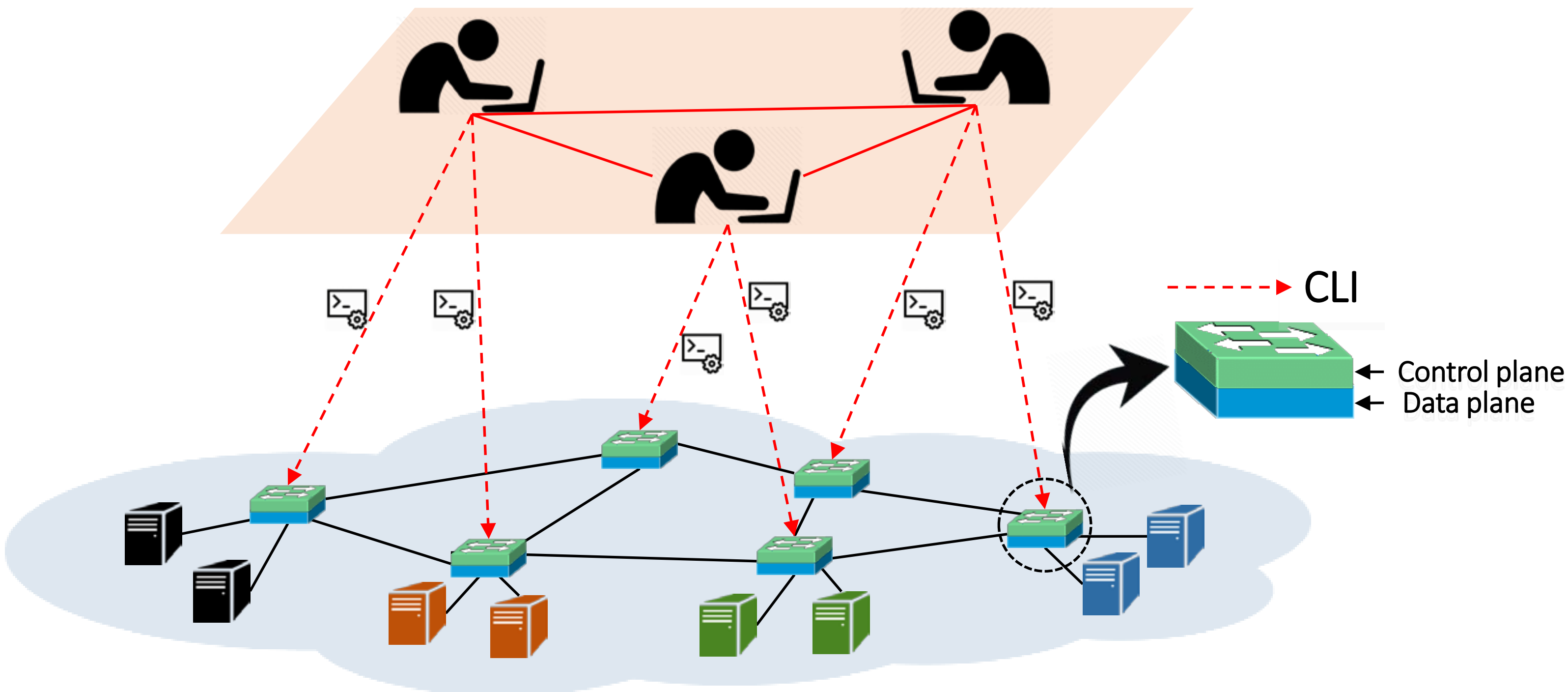
Outline

- Introduction
- SDN-RBAC Model
- Parameterized Permissions and Roles
- ParaSDN Model for Fine Grained and Scalable Authorization in SDN
- SDN-RBACa Administrative Model
- Proxy Operations and Custom Permissions
- Conclusion and Future Work

Introduction
Traditional Networks

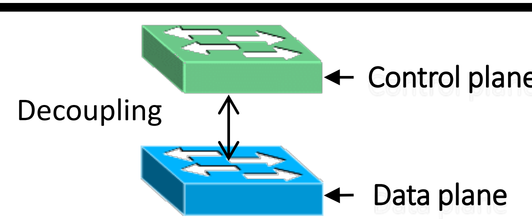
Management Layer

Infrastructure Layer

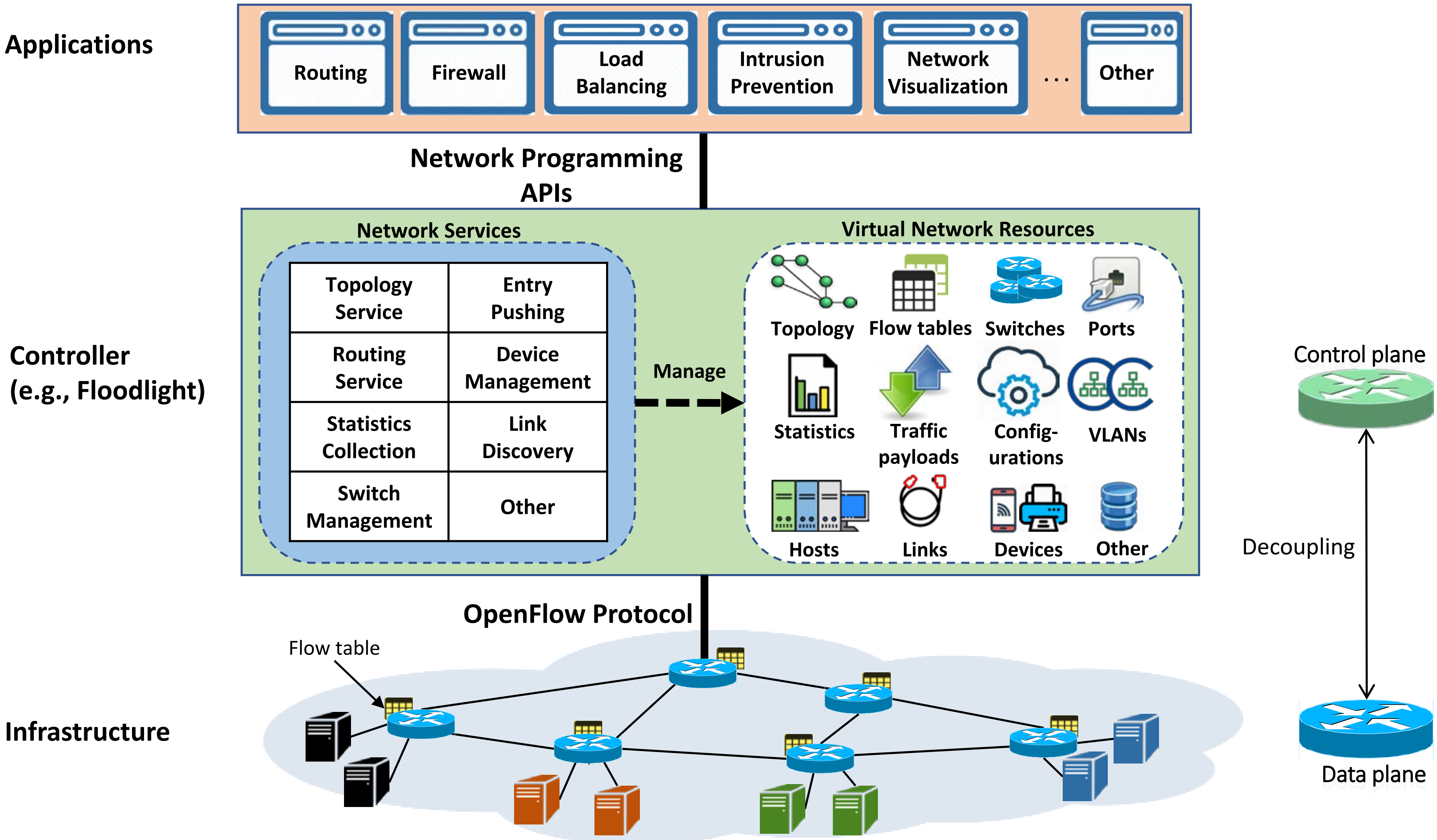


Traditional Network

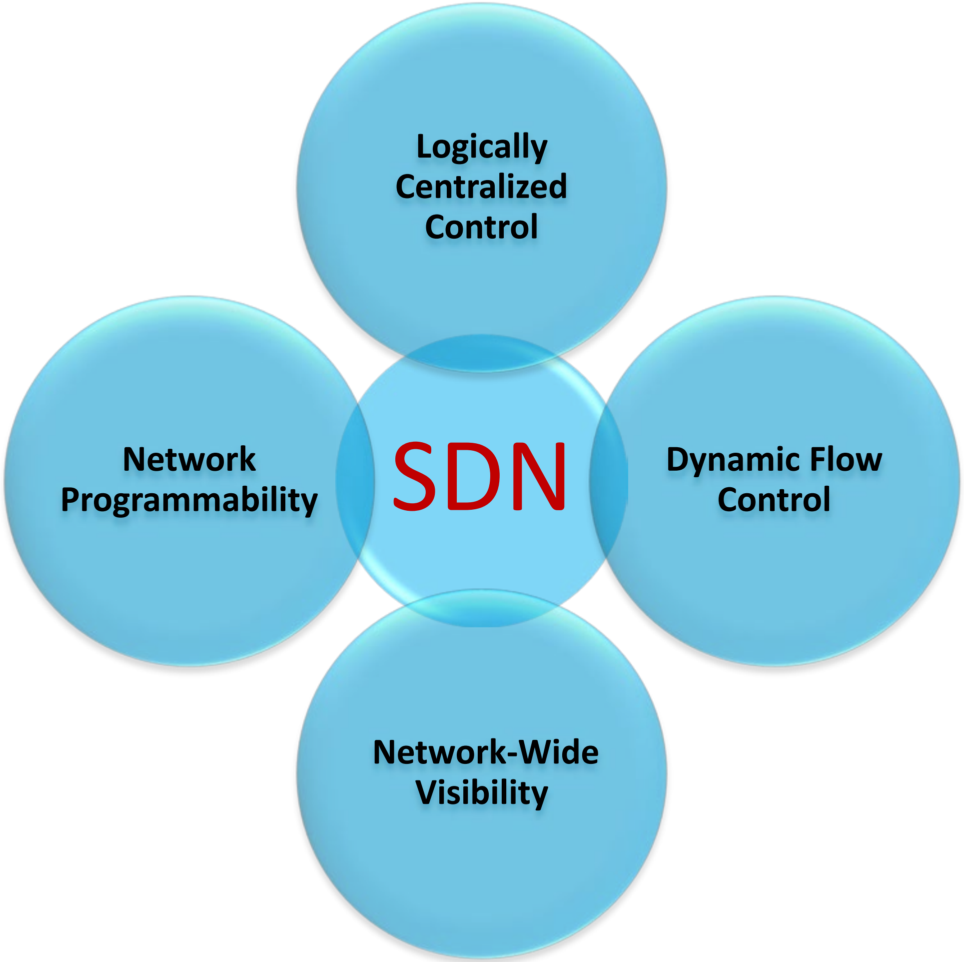
SDN Idea



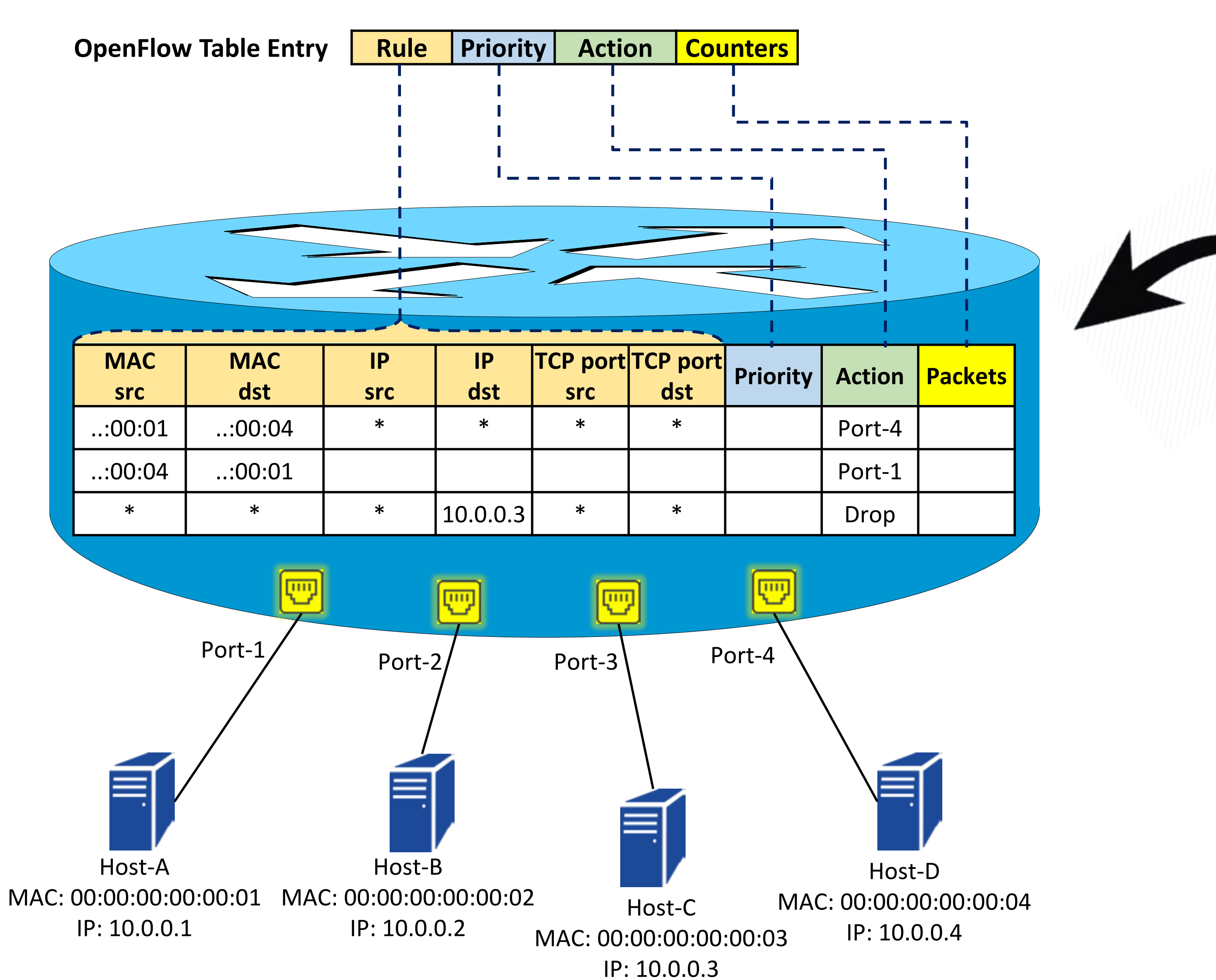
Introduction
Software Defined Networks (SDN)



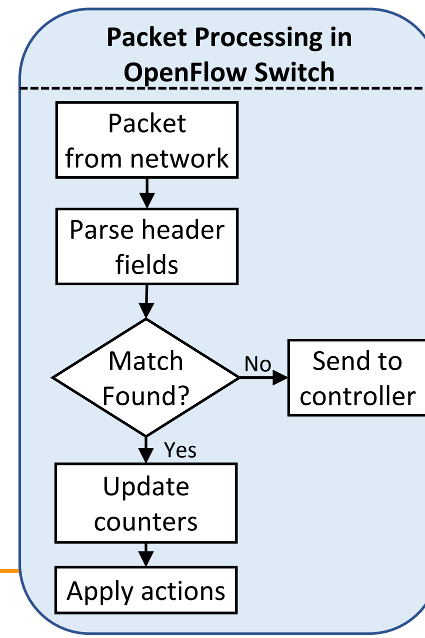
Features Provided by SDN Architecture



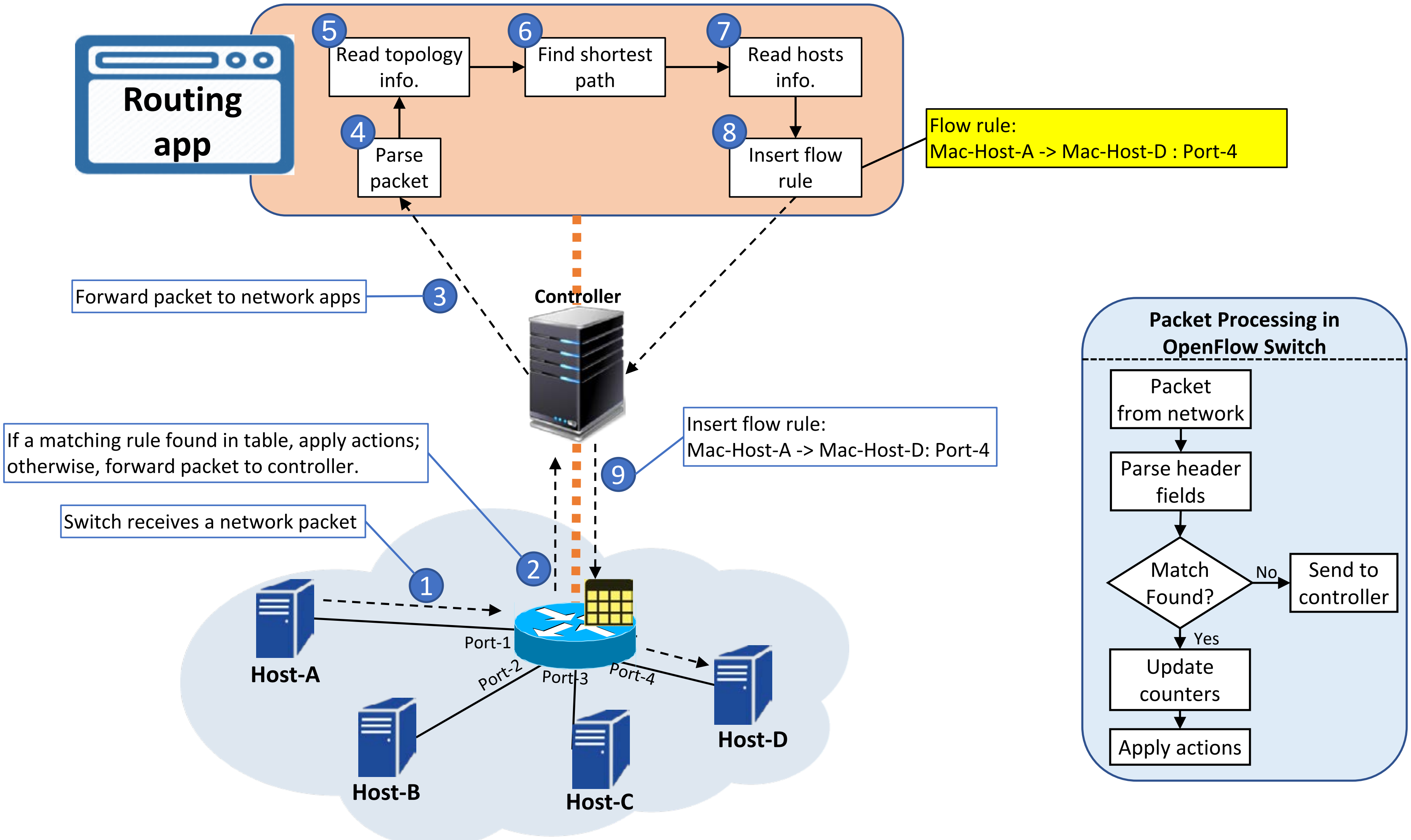
Flow Table Structure



MAC src	MAC dst	IP src	IP dst	TCP port src	TCP port dst	Priority	Action	Packets
..:00:01	..:00:04	*	*	*	*		Port-4	
..:00:04	..:00:01						Port-1	
*	*	*	10.0.0.3	*	*		Drop	

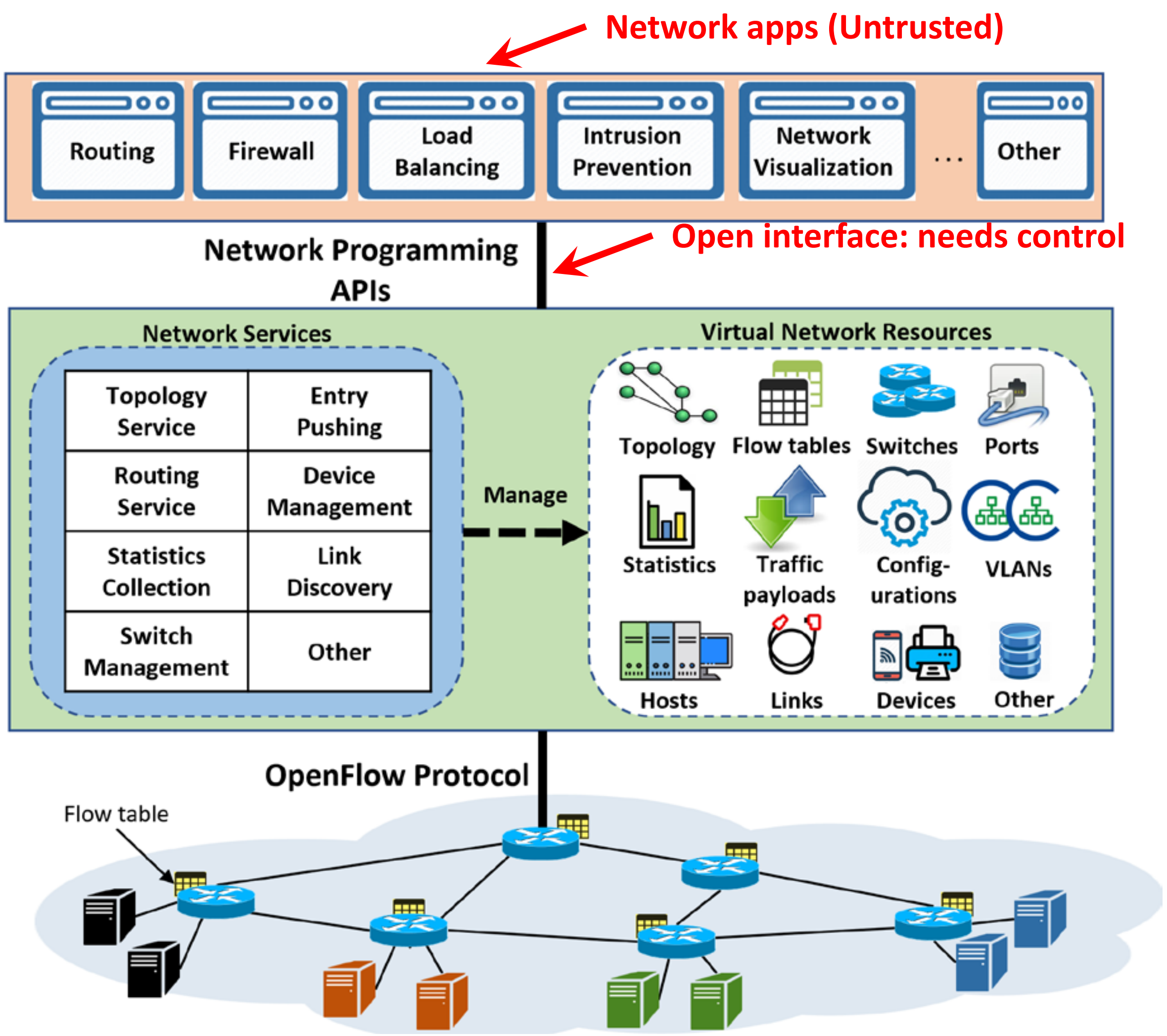


Flow Rule Insertion Example



Access Control in SDN

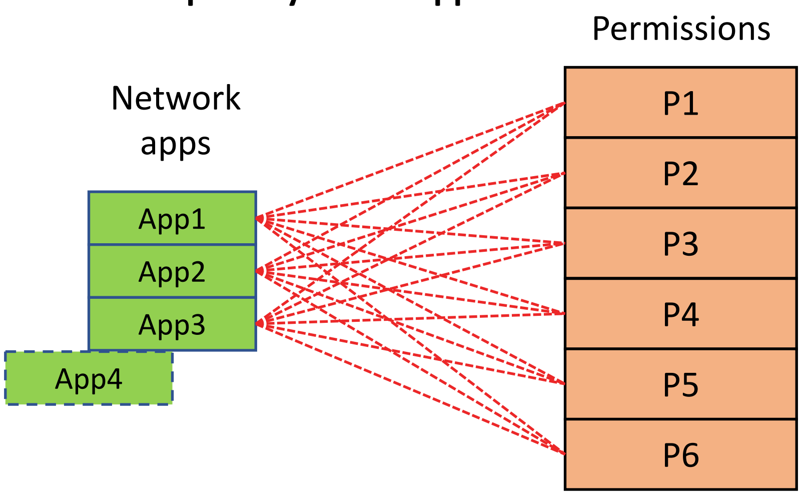
- Control which subjects (network apps) can access which objects (virtual network resources) for performing which actions (SDN operations).



Literature of Access Control for SDN

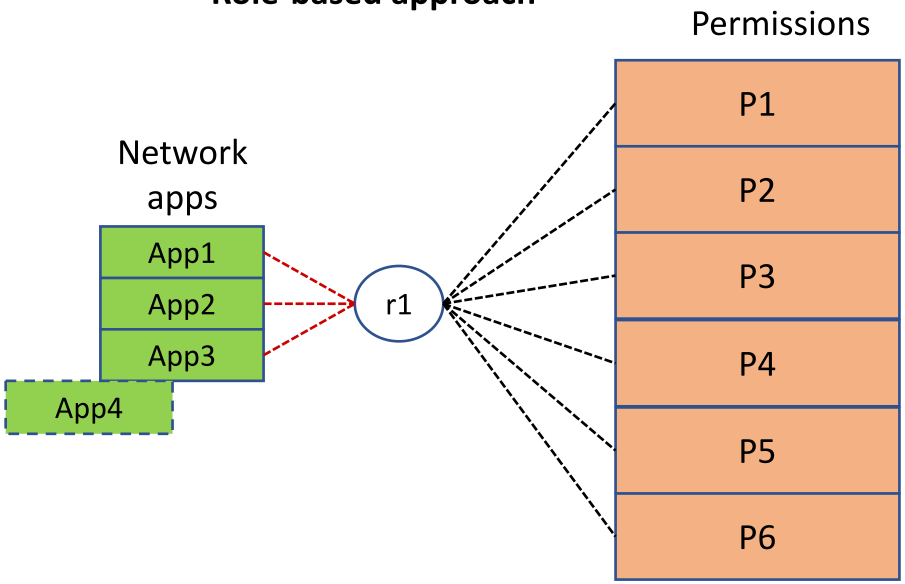
- Capability-based approaches
  - Direct relation between operations and apps.
  - Well studied and known to have administrative complexities.

Capability-based approach



Total associations =3 x 6 = 18
1 new app requires 6 new associations
1 new permission requires 3 new associations

Role-based approach



Total associations = 3 + 6 = 9
1 new app requires 1 new associations
1 new permission requires 1 new association

Problem and Thesis Statement

Problem Statement:

Current Software Defined Networking technology is lacking access control models and enforcement for protecting network resources residing in the SDN controller from unauthorized access by OpenFlow applications.

Thesis Statement:

Role-based access control model and its extensions is an effective approach for the specification and administration of dynamic access control for Software Defined Networking.

Summary of Contributions

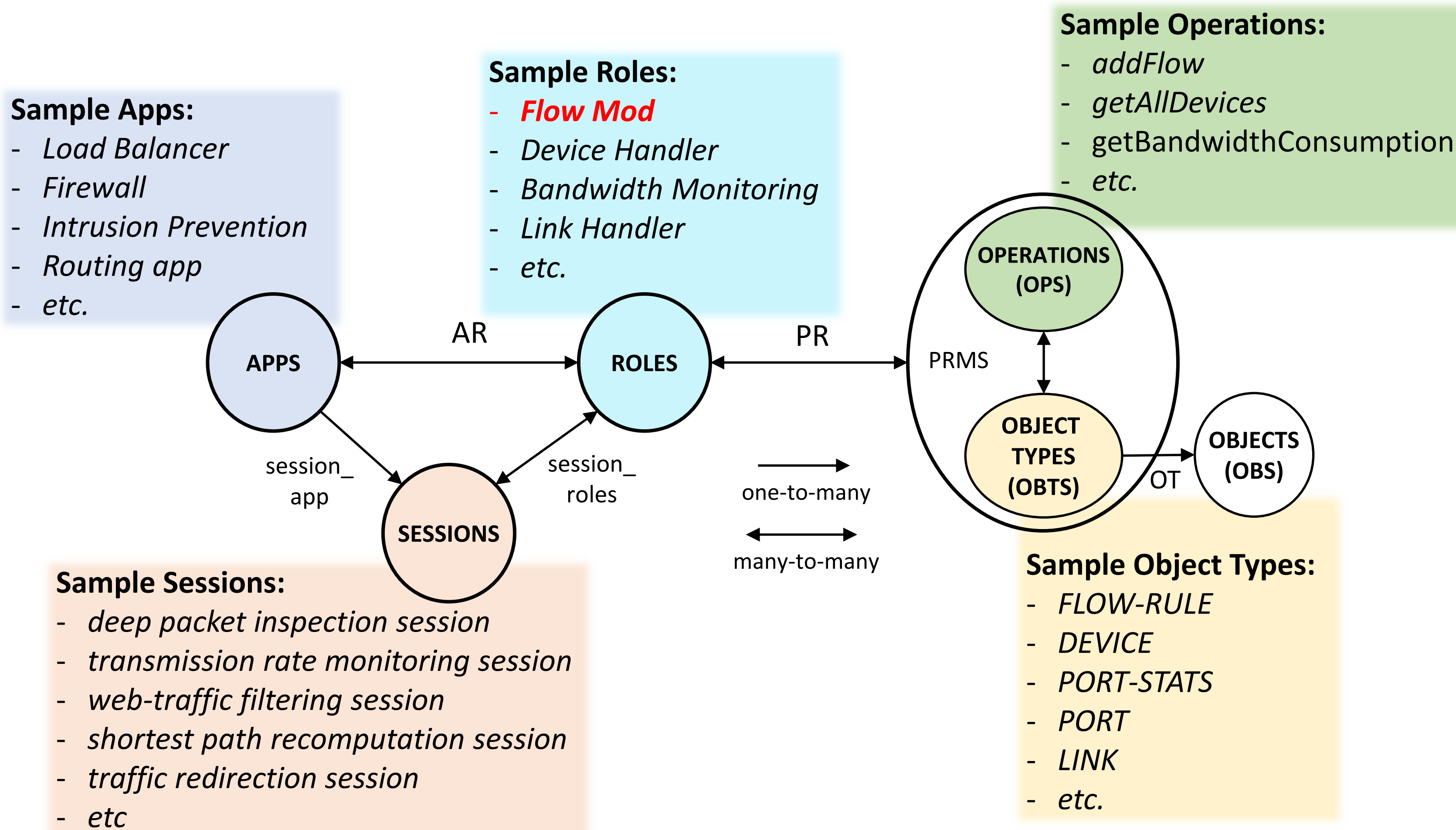
- **Enabling Role Based Authorization for SDN.**
  - SDN-RBAC Model and Authorization Framework with Implementation & Enforcement in SDN Controller.
- **Fine-Grained and Scalable Access Control for SDN.**
  - Access Control Enhanced with Role and Permission Parameters with Authorization Framework Extended with Parameter Engine and Enforcement in SDN Controller.
- **Administration of Access Control in SDN.**
  - SDN-RBACa Administrative Model for Managing roles, Permissions and Network App Authorizations in SDN.
  - Proxy Operations and Custom Permissions for Enhanced Engineering of Administrative Units in SDN.

Summary of Contributions

- **Enabling Role Based Authorization for SDN.**
  - SDN-RBAC Model and Authorization Framework with Implementation & Enforcement in SDN Controller.
- **Fine-Grained and Scalable Access Control for SDN.**
  - Access Control Enhanced with Role and Permission Parameters with Authorization Framework Extended with Parameter Engine and Enforcement in SDN Controller.
- **Administration of Access Control in SDN.**
  - SDN-RBACa Administrative Model for Managing roles, Permissions and Network App Authorizations in SDN.
  - Proxy Operations and Custom Permissions for Enhanced Engineering of Administrative Units in SDN.

SDN-RBAC: Conceptual Model

- *Design goal*: conformance with the standard NIST-RBAC Reference Model.
- SDN-RBAC adopts standard RBAC model with evolutionary changes, rather than revolutionary.



SDN-RBAC Formal Model Definition

1. Model Element Sets:

- $APPS$, $ROLES$, OPS, OBS and $OBTS$, a finite set of OpenFlow apps, roles, operations, objects and object types, respectively.
- $PRMS \subseteq 2^{OPS \times OBTS}$, the set of permissions.
- $SESSIONS$, a finite set of sessions.

2. Assignment Relations:

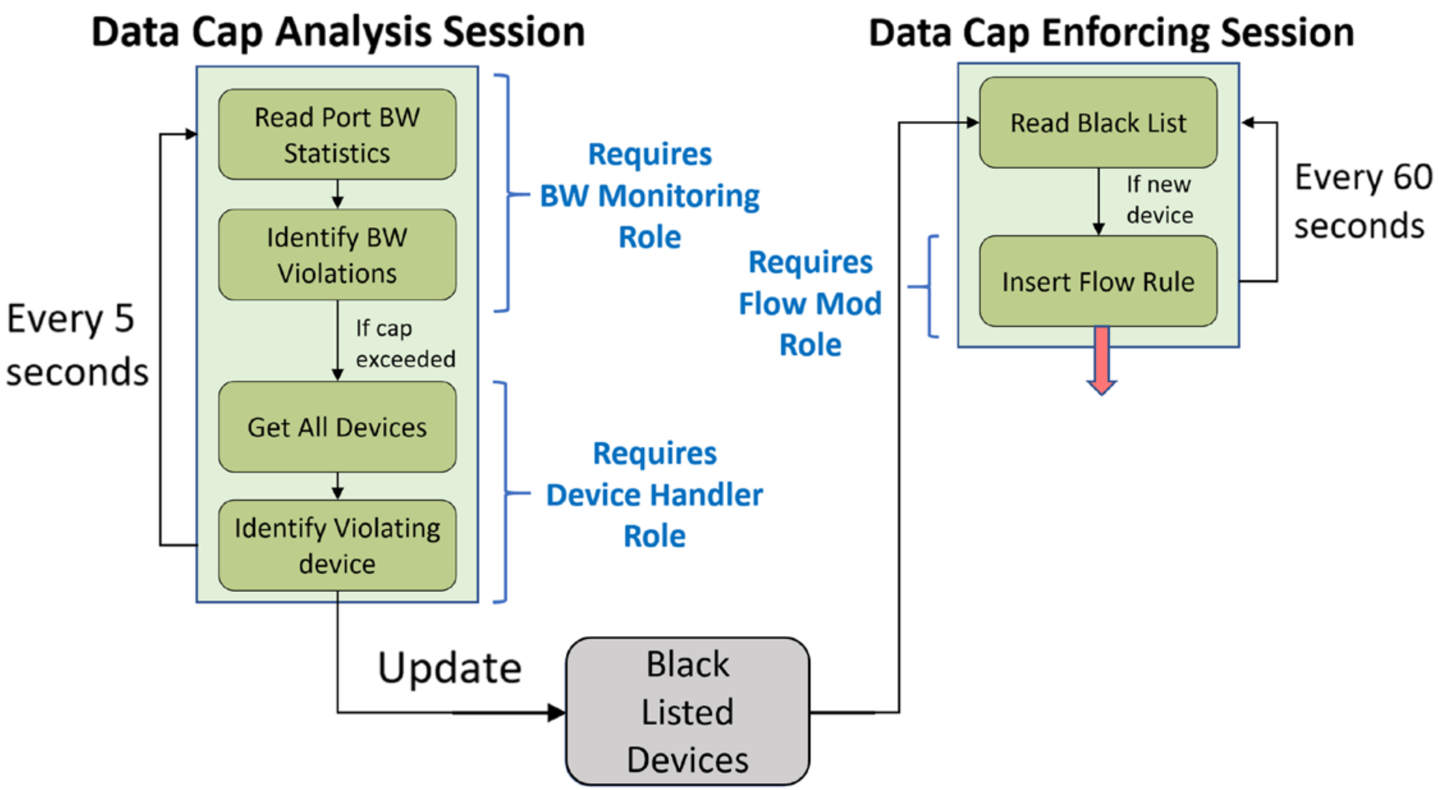
- $PR \subseteq PRMS \times ROLES$, a many-to-many mapping permission-to-role assignment relation.
- $AR \subseteq APPS \times ROLES$, a many-to-many mapping app-to-role assignment relation.
- $OT \subseteq OBS \times OBTS$, a many-to-one relation mapping an object to its type.

3. Derived 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 $(o, t_1) \in OT \wedge (o, t_2) \in OT \Rightarrow t_1 = t_2$.
- $\boxed{avail\_session\_perms(s : SESSIONS)} \rightarrow 2^{PRMS}$, the permissions available to an app in a session $= \bigcup_{r \in session\_roles(s)} assigned\_perms(r)$.

Use-case in SDN-RBAC

Multi session app: Data Usage Cap Manager



Use-Case Security Configuration in SDN-RBAC

1. Use-Case Sets:

- APPS = {DataUsageCapMngr}
- ROLES = {Device Handler, Bandwidth Monitoring, Flow Mod}.
- OBS = D ∪ FR ∪ PS, where D = set of all network devices, FR = set of all flow rules, and PS = set of all port statistics in all switches.
- OBTS = {DEVICE, PORT-STATS, FLOW-RULE}.
- PRMS = {(getAllDevices, DEVICE), (getBandwidthConsumption, PORT-STATS), (addFlow, FLOW-RULE)}.
- SESSIONS = {DataUsageAnalysisSession, DataCapEnforcingSession}.

2. Assignment Relations:

- PR = {((getAllDevices, DEVICE), Device Handler), ((getBandwidthConsumption, PORT-STATS), Bandwidth Monitoring), ((addFlow, FLOW-RULE), Flow Mod)}
- AR = {(DataUsageCapMngr, Device Handler), (DataUsageCapMngr, Bandwidth Monitoring), (DataUsageCapMngr, Flow Mod)} .
- OT = {(d, DEVICE) : d ∈ D} ⋃ {(ps, PORT-STATS) : ps ∈ PS} ⋃ {(fr, FLOW-RULE) : fr ∈ FR}.

3. Derived Functions:

- assigned_perms(DeviceHandler) = {(getAllDevices, DEVICE)}, assigned_perms(BandwidthMonitoring) = {(getBandwidthConsumption, PORT-STATS)}, assigned_perms(FlowMod) = {(addFlow, FLOW-RULE)}.
- app_sessions(DataUsageCapMngr) = {DataUsageAnalysisSession, DataCapEnforcingSession}.
- session_app(DataUsageAnalysisSession) = {DataUsageCapMngr}. session_app(DataCapEnforcingSession) = {DataUsageCapMngr}.
- session_roles(DataUsageAnalysisSession) = {Device Handler, Bandwidth Monitoring}. session_roles(DataCapEnforcingSession) = {Flow Mod}.
- avail_session_perms(DataUsageAnalysisSession) = {(getAllDevices, DEVICE), (getBandwidthConsumption, PORT-STATS)}. avail_session_perms(DataCapEnforcingSession) = {(addFlow, FLOW-RULE)}.

3 roles

2 sessions

permission to insert flow rules

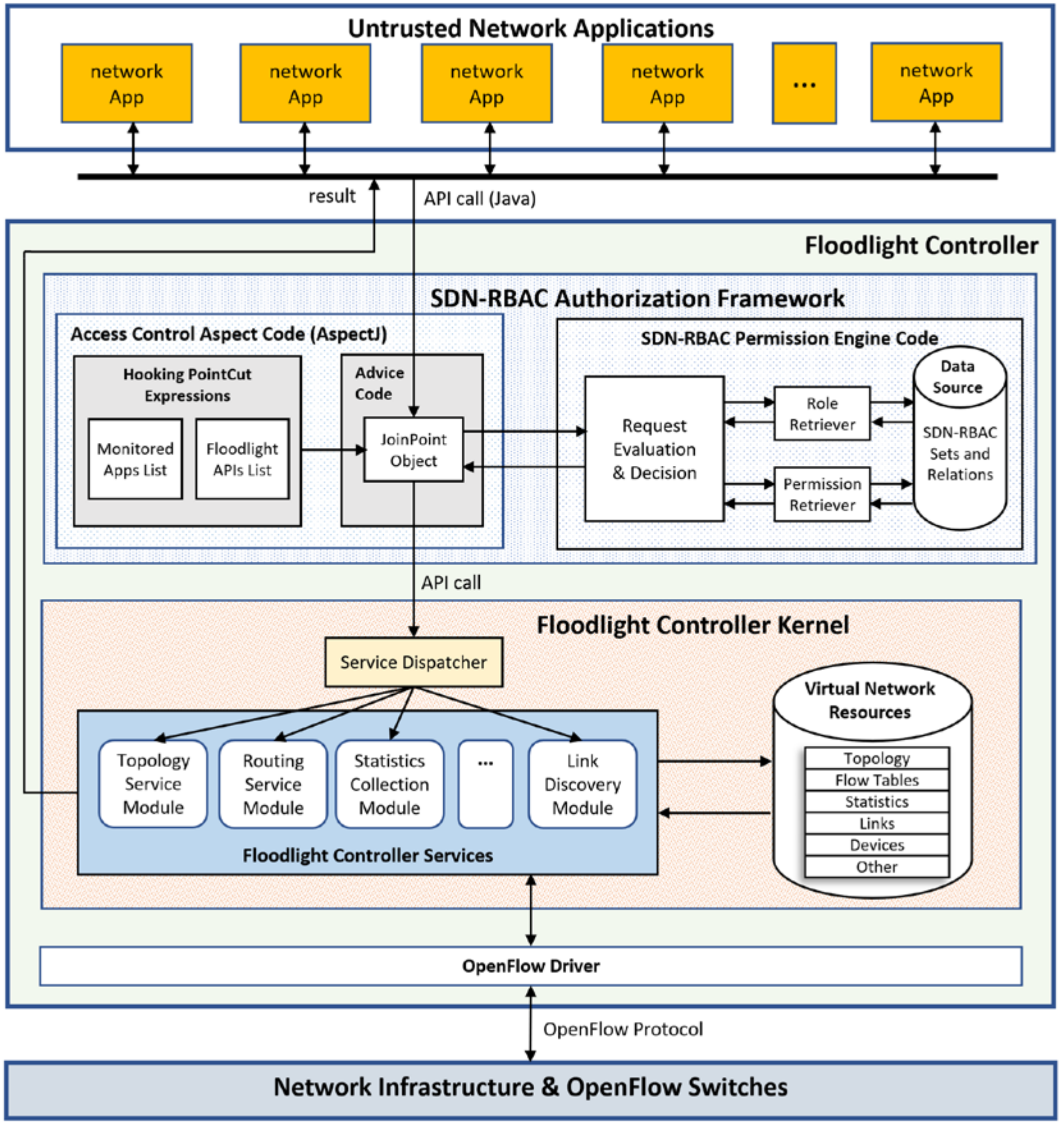
role assigned to app

very important role & permission

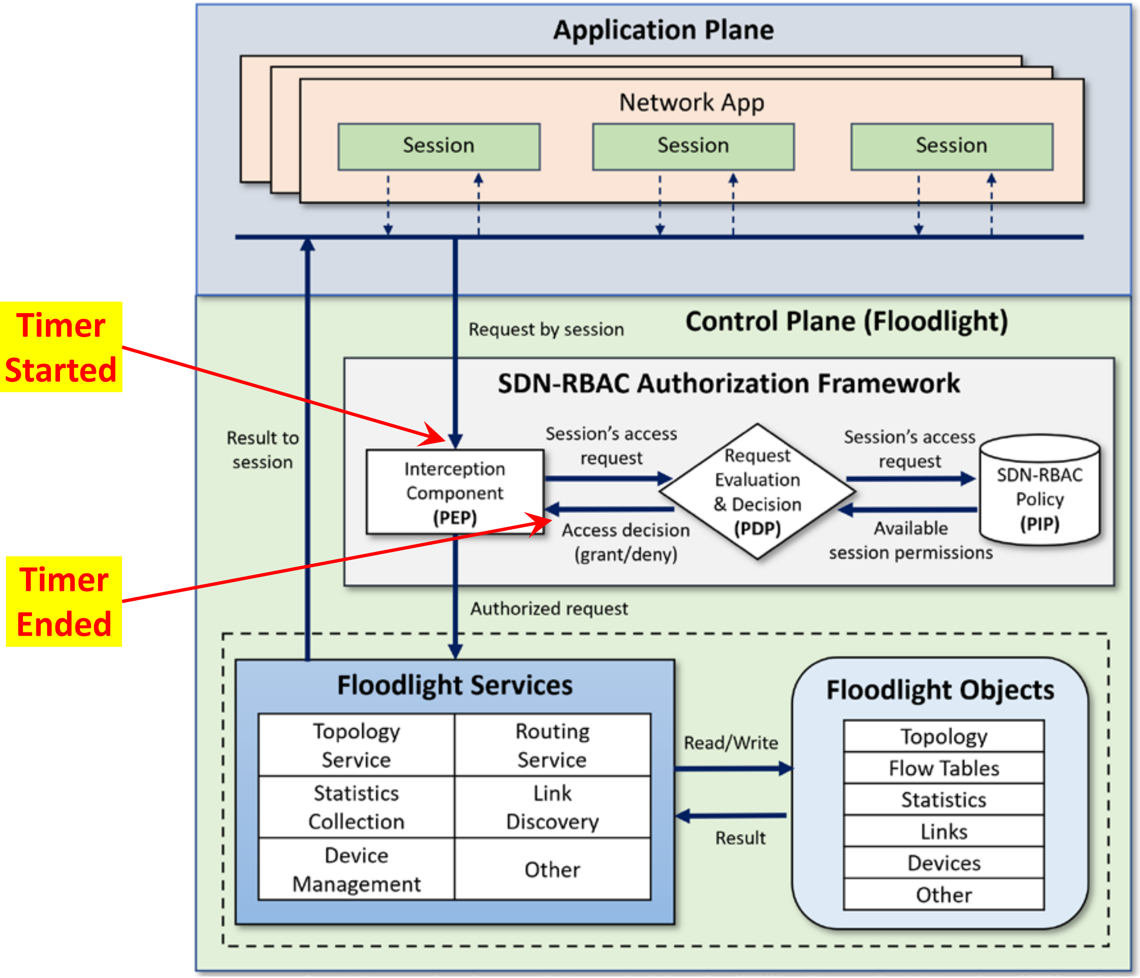
role activated in session

permission available to session

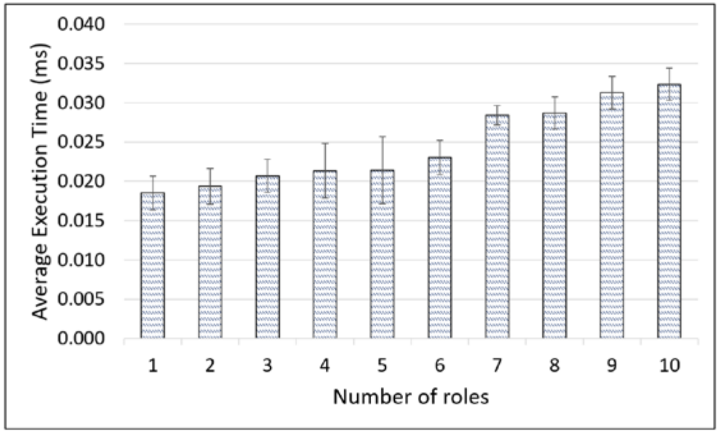
SDN-RBAC Framework Implementation in Floodlight



SDN-RBAC Average Authorization Time



- Test app with 50 ops covered by 10 different roles.
- Report authorization time for all 50 requests.
- Different security policies.
- Test repeated 100 times for each security policy.
- Average authorization time is calculated.
- floodlight's boot-up time is ignored.



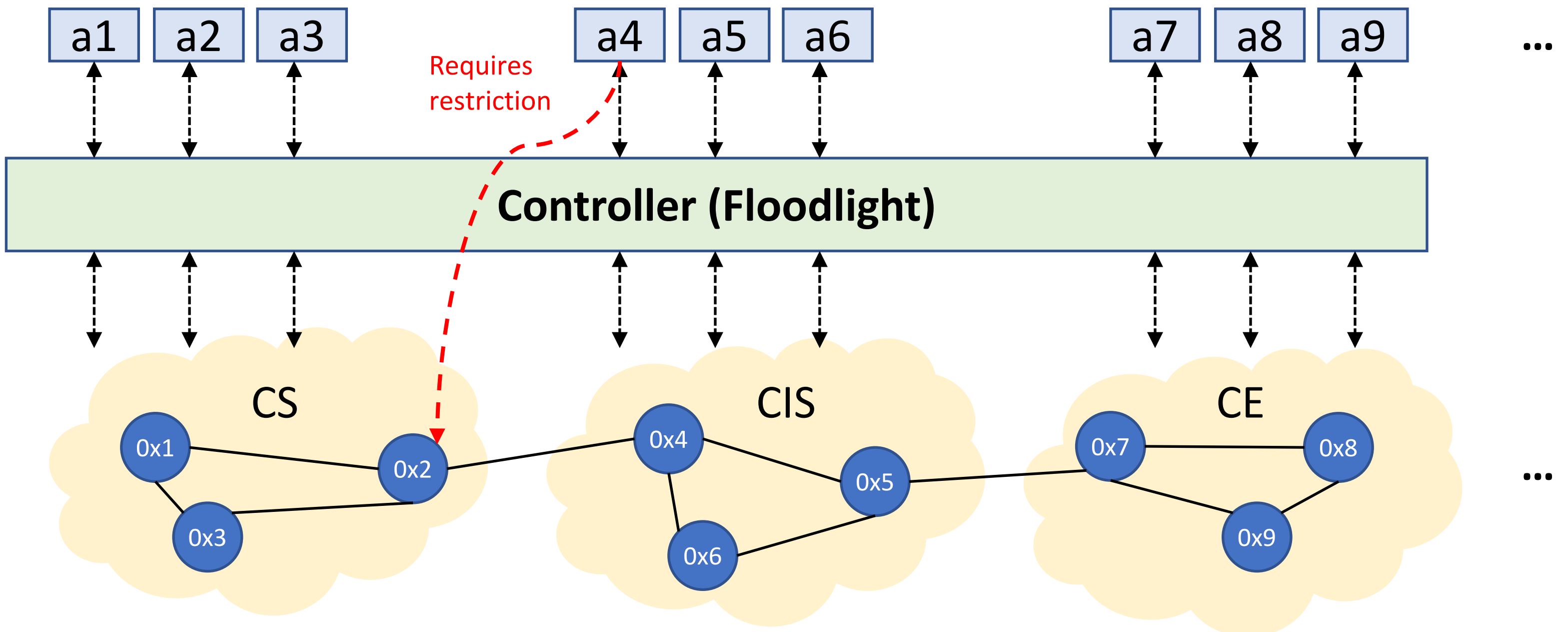
On average: 0.0245 ms overhead for 50 operations.

Summary of Contributions

- **Enabling Role Based Authorization for SDN.**
  - SDN-RBAC Model and Authorization Framework with Implementation & Enforcement in SDN Controller.
- **Fine-Grained and Scalable Access Control for SDN.**
  - Access Control Enhanced with Role and Permission Parameters with Authorization Framework Extended with Parameter Engine and Enforcement in SDN Controller.
- **Administration of Access Control in SDN.**
  - SDN-RBACa Administrative Model for Managing roles, Permissions and Network App Authorizations in SDN.
  - Proxy Operations and Custom Permissions for Enhanced Engineering of Administrative Units in SDN.

Limitations of SDN-RBAC

- Apps are authorized on object types (e.g., (addFlow, FLOW RULE)) → Fine grained access control is required.



Role: Flow Mod$_1$
Assigned Perms:
(addFlow, flow_rule^{sw0x1})
(addFlow, flow_rule^{sw0x2})
(addFlow, flow_rule^{sw0x3})

Role: Flow Mod$_2$
Assigned Perms:
(addFlow, flow_rule^{sw0x4})
(addFlow, flow_rule^{sw0x5})
(addFlow, flow_rule^{sw0x6})

Role: Flow Mod$_3$
Assigned Perms:
(addFlow, flow_rule^{sw0x7})
(addFlow, flow_rule^{sw0x8})
(addFlow, flow_rule^{sw0x9})

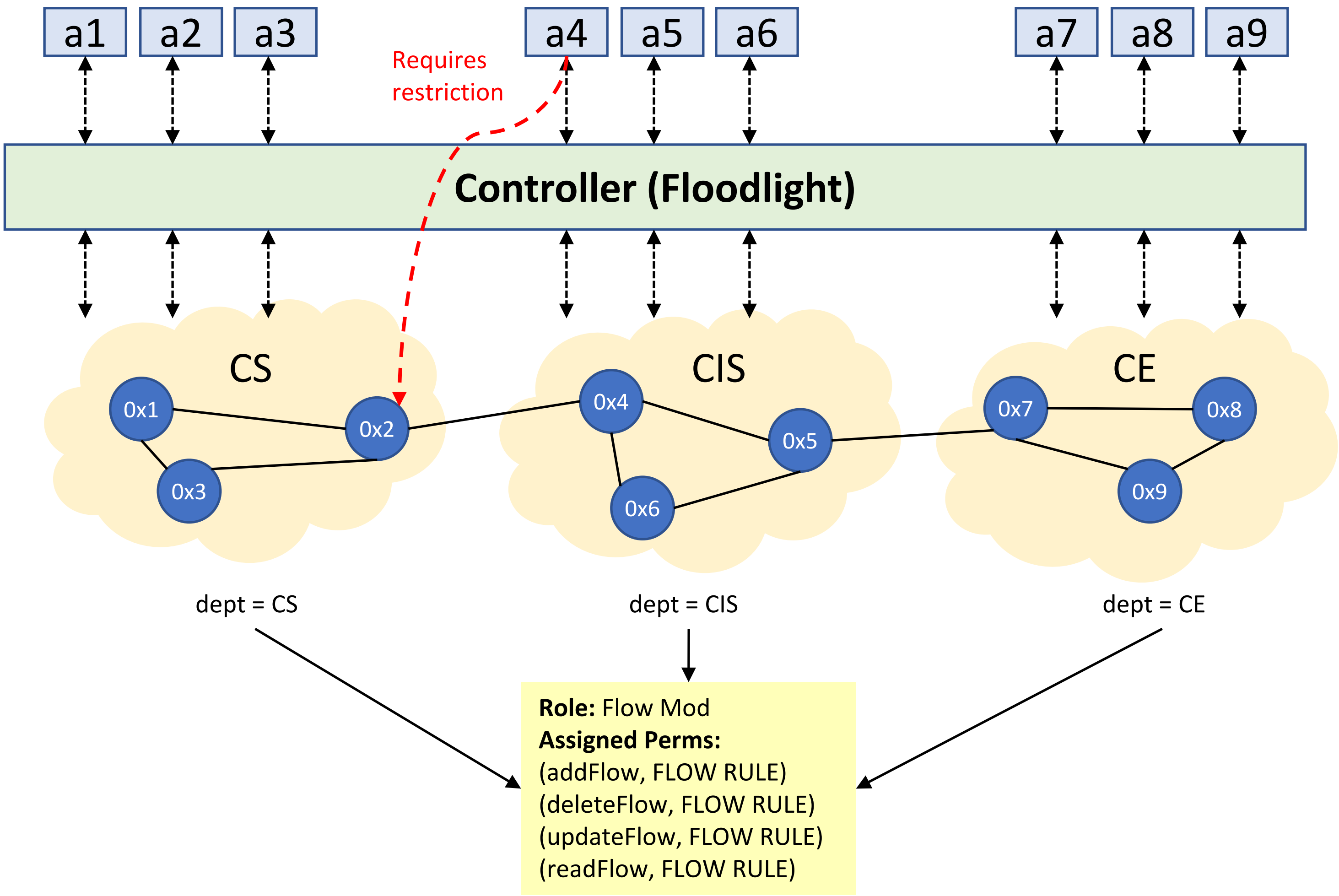
...

- Multiple very closely related roles are defined to achieve fine-grained access control.
- Roles are limited in membership.

Role explosion

Permission explosion

Introducing Parameterized Roles and Permissions in SDN



Parameterized Permissions and Roles

- **Parameters**
  - name:value pairs.
  - Add restrictions on access to network resources.

- **Parameterized Roles:**

$$(r_i, \{(par_1, val_1), (par_2, val_2), \ldots\})$$

Example:

$$(\text{Flow Mod}, \{(\text{dept}, \perp), (\text{traffic}, \perp)\})$$

- **Parameterized Permissions:**

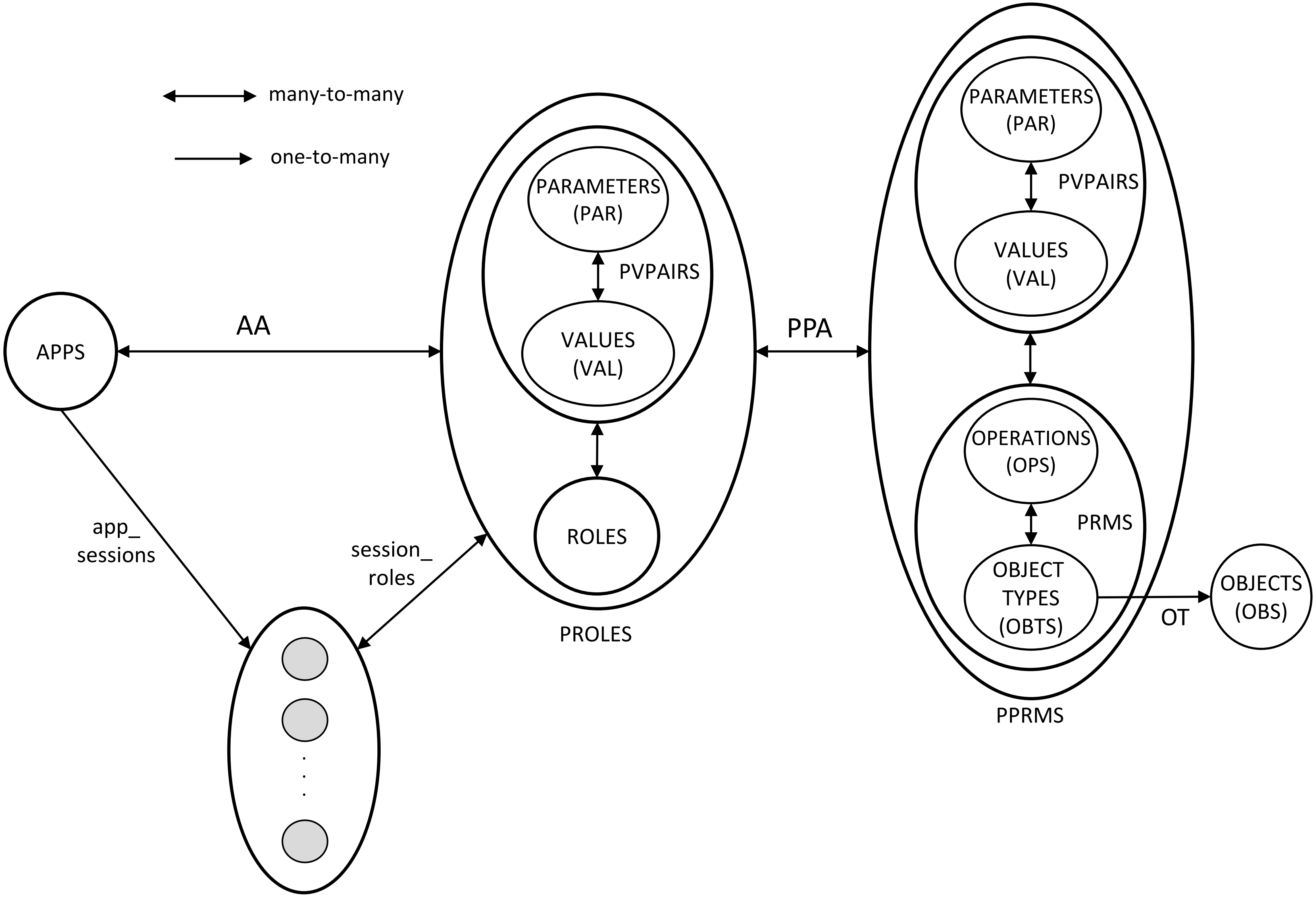
$$((op_i, ot_i), \{(par_1, val_1), (par_2, val_2), \ldots\})$$

Example:

$$((\text{addFlow}, \text{FLOW-RULE}), \{(\text{dept}, \perp), (\text{traffic}, \perp)\})$$

$\perp$ = *Unknown.*

ParaSDN Conceptual Model



ParaSDN Formal Model Definition

1.Basic Sets:

- APPS, ROLES, OPS, OBS, OBTS, PAR, and VAL: set of apps, roles, operations, objects, object types, parameters, and parameter values.
- For each *par* $\in$ PAR, Range(*par*) represents the parameter's range, a finite set of atomic values. We assume VAL includes a special value "$\perp$" to indicate that the value of a parameter is unknown.
- parType: PAR $\rightarrow$ {set, atomic} specifies parameter type as set of atomic valued.
- PRMS $\subseteq$ OPS $\times$ OBTS, set of ordinary permissions.
- SESSIONS, set of sessions.

2.Assignment Relations:

- OT $\subseteq$ OBS $\times$ OBTS, a many-to-one relation mapping an object to its type, where $(o, ot_1) \in \mathrm{OT} \wedge (o, ot_2) \in \mathrm{OT} \Rightarrow ot_1 = ot_2$.
- PVPAIRS $\subseteq$ PAR $\times$ VAL, a many-to-many mapping parameter to value assignment relation. For convenience, for every pvpair = (par$_i$, val$_i$) $\in$ PVPAIRS, let pvpair.par = par$_i$ and pvpair.val = val$_i$.
- PPRMS $\subseteq$ PRMS $\times$ $2^{PVPAIRS}$, a relation mapping a permission role to subset of (parameters , value) combinations. For convenience, for every pp = ((op$_i$, ot$_i$), PVPAIRS$_i$) $\in$ PPRMS, let pp.op = op$_i$, pp.ot = ot$_i$, and pp.PVPAIRS = PVPAIRS$_i$.
- PROLES $\subseteq$ ROLES $\times$ $2^{PVPAIRS}$, a relation mapping a role to subset of combinations of parameters and their values. For convenience, for every pr = (r$_i$, PVPAIRS$_i$) $\in$ PROLES, let pr.r = r$_i$ and pr.PVPAIRS = PVPAIRS$_i$.
- PPA $\subseteq$ PPRMS $\times$ PROLES, a many-to-many mapping parameterized permission to parameterized role assignment relation.
- AA $\subseteq$ APPS $\times$ PROLES, a many-to-many mapping app to parameterized role assignment relation.

3.Derived Functions:

- assigned_pperms: PROLES $\rightarrow$ 2^{PPRMS}, the mapping of parameterized role into a set of parameterized permissions. Formally, assigned_pperms(pr) = {pp $\in$ PPRMS — (pp, pr) $\in$ PPA}.
- app_sessions: APPS $\rightarrow$ $2^{SESSIONS}$, the mapping of an app into a set of sessions.
- session_app : SESSIONS $\rightarrow$ 2^{APPS}, the mapping of session into the corresponding app.
- session_roles: SESSIONS $\rightarrow$ 2^{PROLES}, the mapping of session into a set of parameterized roles. Formally, session_roles(s) = {pr $\in$ PROLES — (session_app(s), pr) $\in$ AA}.
- type: OBS $\rightarrow$ OBTS, a function specifying the type of an object defined as type(o) = $\{t \in \mathrm{OBTS}$ — $(o, t) \in \mathrm{OT}\}$.
- avail_session_pperms: SESSIONS $\rightarrow$ 2^{PPRMS}, the parameterized permissions available to an app in a session. Formally, avail_session_pperms(s) = $\bigcup_{pr \in session_roles(s)}$assigned_pperms(pr).

4.Parameter Verification Functions:

- VERIFIERS = {V$_1$, V$_2$, ..., V$_n$} a finite set of Boolean functions. For each V$_i$ $\in$ VERIFIERS.V$_i$: SESSIONS $\times$ OPS $\times$OBS $\times$ PVPAIRS $\rightarrow$ {True, False}.
- param_verifier: OBTS $\times$ PAR $\rightarrow$ VERIFIERS, a function that maps a combination of object type and parameter to the corresponding verification function needs to be evaluated.

Use-Case Security Configuration in SDN-RBAC

1. Model Basic Sets:

- APPS = {Data Usage Cap Mngr, Intrusion Prevention App}.
- ROLES = {Device Handler, Bandwidth Monitoring, Flow Mod, Packet-In Handler}.
- OPS = {queryDevice, getBandwidthConsumption, addFlow, readPacketInPayload}.
- OBS = D $\cup$ PS $\cup$ FR $\cup$ PIP, where D = set of all network devices, PS = set of all port statistics in all switches, FR = set of all flow rules, and PIP = set of all packet-in messages.
- OBTS = {DEVICE, PORT-STATS, FLOW-RULE, PI-PAYLOAD}.
- PAR = {vlan_id, attachment_point, dept, traffic}.
- Range(vlan_id) = {1, 2}. Range(attachment_point) = {0x1:1, 0x1:2, 0x2:1, 0x2:2, 0x3:1}. Range(dept) = {CS, CE}. Range(traffic) = {web}.
- parType(vlan_id) = atomic. parType(attachment_point) = set. parType(dept) = set. parType(traffic) = atomic.
- PRMS = {(queryDevice, DEVICE), (getBandwidthConsumption, PORT-STATS), (addFlow, FLOW-RULE), (readPacketInPayload, PI-PAYLOAD)}.
- SESSIONS = {DataUsageAnalysisSession, DataCapEnforcingSession, IntrusionPreventionSession}.

2. Assignment Relations:

- OT = {(d, DEVICE) : d $\in$ D} $\bigcup$ {(ps, PORT-STATS) : ps $\in$ PS} $\bigcup$ {(fr, FLOW-RULE) : fr $\in$ FR} $\bigcup$ {(pip, PI-PAYLOAD) : pip $\in$ PIP}}.
- PPRMS = {((queryDevice, DEVICE), {(vlan_id, $\perp$)}), ((getBandwidthConsumption, PORT-STATS), {(attachment_point, $\perp$)}), ((addFlow, FLOW-RULE), {(dept, $\perp$), (traffic, $\perp$)}), ((readPacketInPayload, PI-PAYLOAD), {(attachment_point, $\perp$)})}
- PROLES = {(Device Handler, {(vlan_id, $\perp$)}), (Bandwidth Monitoring, {(attachment_point, $\perp$)}), (Flow Mod, {(dept, $\perp$), (traffic, $\perp$)}), (Packet-In Handler, {(attachment_point, $\perp$)})}
- PPA = {(((queryDevice, DEVICE), {(vlan_id, $\perp$)}), (Device Handler, {(vlan_id, $\perp$)})), (((getBandwidthConsumption, PORT-STATS), {(attachment_point, $\perp$)}), (Bandwidth Monitoring , {(attachment_point, $\perp$)})), (((addFlow, FLOW-RULE), {(dept, $\perp$), (traffic, $\perp$)}), (Flow Mod, {(dept, $\perp$), (traffic, $\perp$)})), (((readPacketInPayload, PI-PAYLOAD), {(attachment_point, $\perp$)}), (Packet-In Handler, {(attachment_point, $\perp$)}))}.
- AA = {(Data Usage Cap Mngr, (Device Handler, {(vlan_id, 1)})), (Data Usage Cap Mngr, (Bandwidth Monitoring, {(attachment_point, {0x1:1, 0x1:2, 0x2:1, 0x2:2})})), (Data Usage Cap Mngr, (Flow Mod, {(dept, {CS}), (traffic, web)})), (Intrusion Prevention App, (Device Handler, {(vlan_id, 2)}), (Intrusion Prevention App, (Packet-In Handler, {(attachment_point, {0x3:1})}), (Intrusion Prevention App, (Flow Mod, {(dept, {CE}), (traffic, web)}))}.

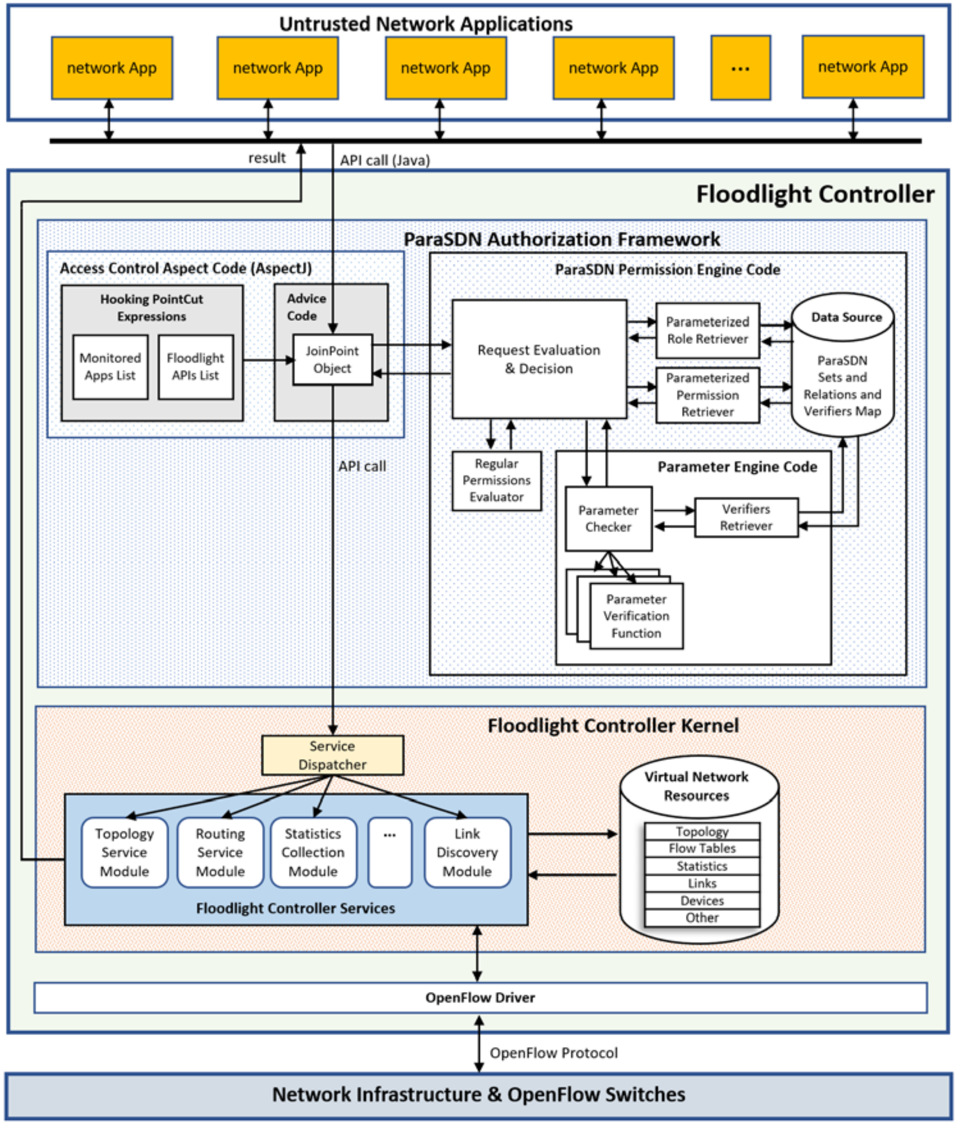
3. Derived Functions:

- assigned_pperms((Device Handler, {(vlan_id, $\perp$)})) = {((queryDevice, DEVICE), {(vlan_id, $\perp$)})}.
  assigned_pperms((Bandwidth Monitoring, {(attachment_point, $\perp$)})) = {((getBandwidthConsumption, PORT-STATS), {(attachment_point, $\perp$)})}.
  assigned_pperms((Flow Mod, {(dept, $\perp$), (traffic, $\perp$)})) = {((addFlow, FLOW-RULE), {(dept, $\perp$), (traffic, $\perp$)})}.
  assigned_pperms((Packet-In Handler, {(attachment_point, $\perp$)})) = {((readPacketInPayload, PI-PAYLOAD), {(attachment_point, $\perp$)})}.
- app_sessions(Data Usage Cap Mngr) = {DataUsageAnalysisSession, DataCapEnforcingSession}.
  app_sessions(Intrusion Prevention App) = {IntrusionPreventionSession}.
- session_roles(DataUsageAnalysisSession) = {(Device Handler, {(vlan_id, 1)}), (Bandwidth Monitoring, {(attachment_point, {0x1:1, 0x1:2, 0x2:1})})}.
  session_roles(DataCapEnforcingSession) = {(Flow Mod, {(dept, {CS}), (traffic, web)})}.
  session_roles(IntrusionPreventionSession) = {(Device Handler, {(vlan_id, 2)}), (Packet-In Handler, {(attachment_point, {0x3:1})}), (Flow Mod, {(dept, {CE}), (traffic, web)})}.
- avail_session_pperms(DataUsageAnalysisSession) = {((queryDevice, DEVICE), {(vlan_id, 1)}), ((getBandwidthConsumption, PORT-STATS), {(attachment_point, {0x1:1, 0x1:2, 0x2:1})})}.
  avail_session_pperms(DataCapEnforcingSession) = {((addFlow, FLOW-RULE), {(dept, {CS}), (traffic, web)})}.
  avail_session_pperms(IntrusionPreventionSession) = {((queryDevice, DEVICE), {(vlan_id, 2)}), ((readPacketInPayload, PI-PAYLOAD), {(attachment_point, {0x3:1})}), ((addFlow, FLOW-RULE), {(dept, {CE}), (traffic, web)})}.

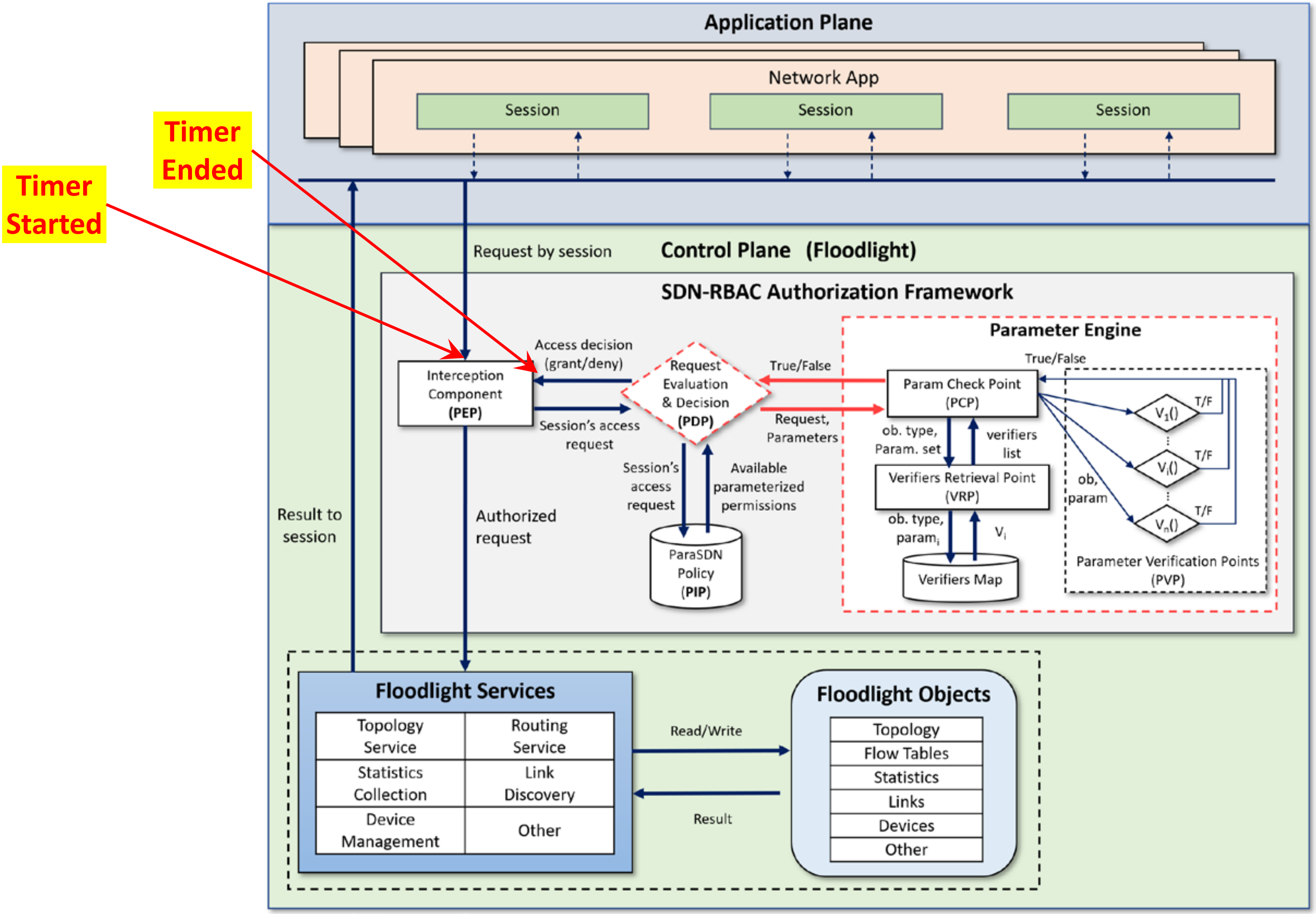
4. Parameter Verification Functions:

- VERIFIERS = {VDeviceVlan, VStatsAttachpoint, VRuleSwitch, VRuleTraffic, VPInAttchpoint}.
- param_verifier((DEVICE, vlan_id)) = VDeviceVlan.
  param_verifier((PORT-STATS, attachment_point)) = VStatsAttachpoint.
  param_verifier((FLOW-RULE, dept)) = VRuleSwitch.
  param_verifier((FLOW-RULE, traffic)) = VRuleTraffic.
  param_verifier((PI-PAYLOAD, attachment_point)) = VPInAttchpoint.

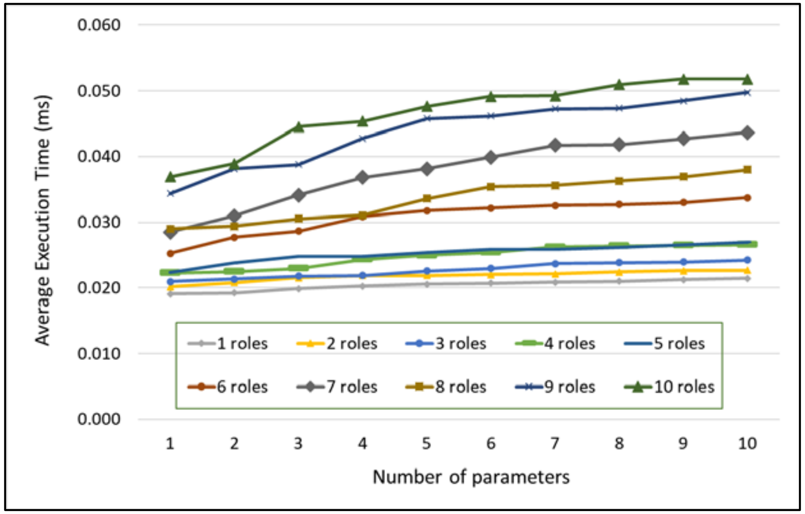
ParaSDN Framework Implementation in Floodlight



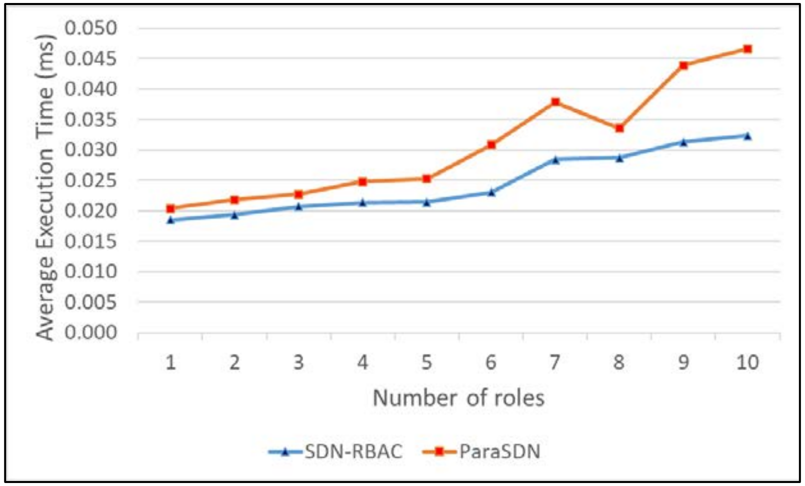
ParaSDN Implementation & Evaluation



ParaSDN Evaluation - 1



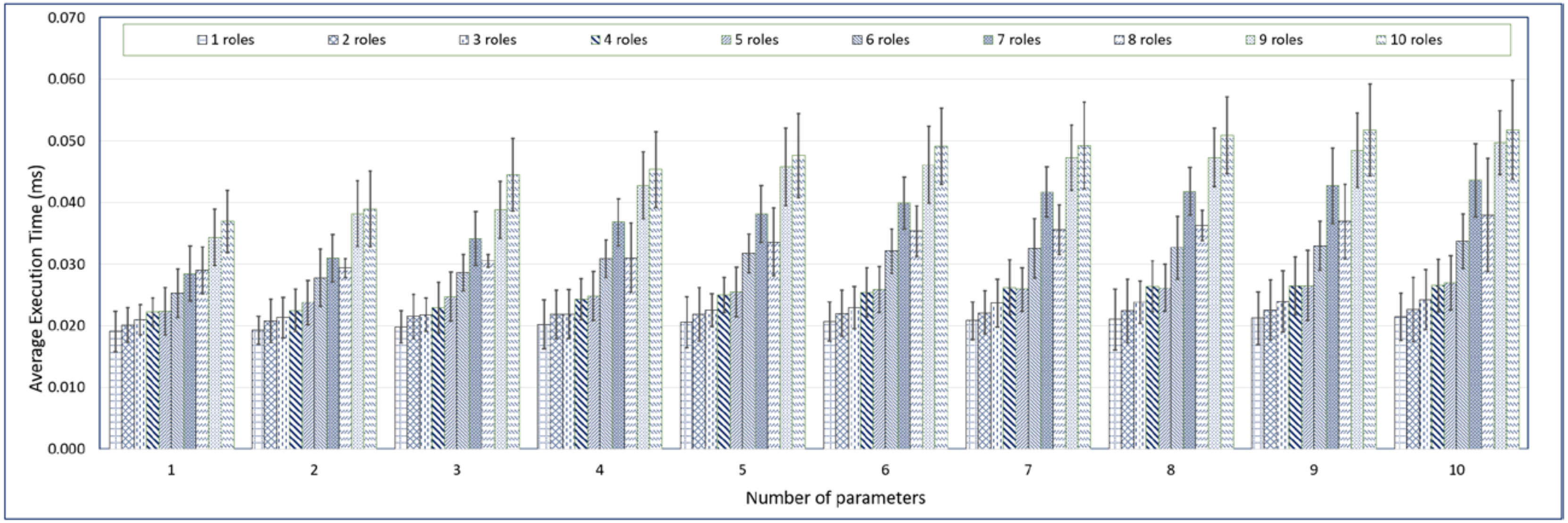
- Test app with 50 ops covered by 10 different roles.
- Report authorization time for all 50 requests.
- Different security policies (parameters and roles).
- Test repeated 100 times for each security policy.
- Average authorization time is calculated.
- Floodlight's boot-up time is ignored.

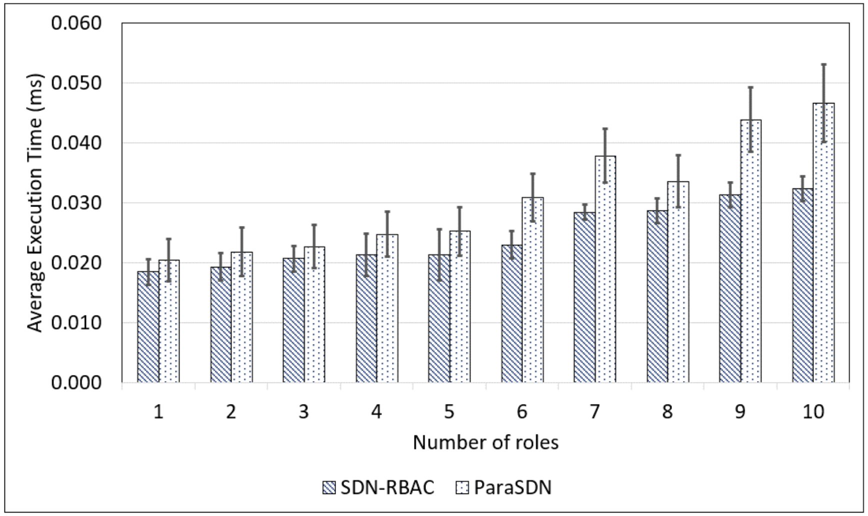


On average: ParaSDN adds 0.031 ms overhead compared to 0.025 for SDN-RBAC.

- 1st parameter in all roles is:
  activePeriod = "08:00-17:00".
- Any request submitted outside active period, will be denied.
- Test 8 is conducted outside active period.

ParaSDN Evaluation - 2



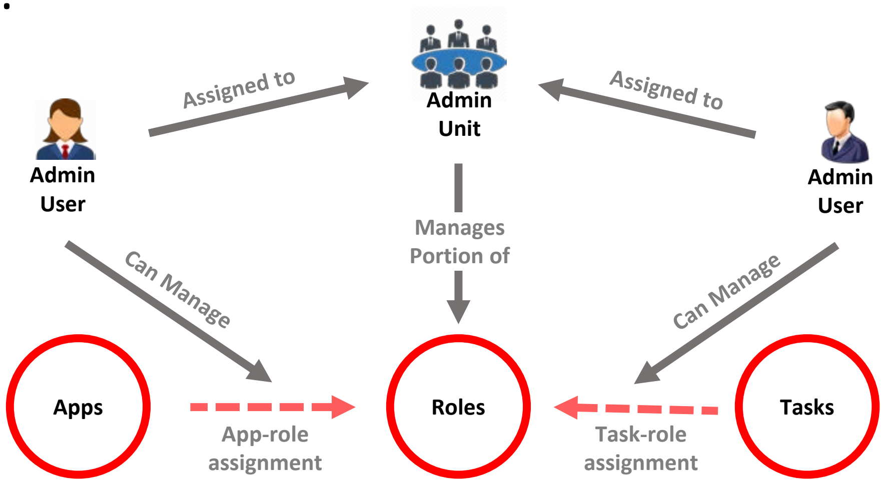


Summary of Contributions

- **Enabling Role Based Authorization for SDN.**
  - SDN-RBAC Model and Authorization Framework with Implementation & Enforcement in SDN Controller.
- **Fine-Grained and Scalable Access Control for SDN.**
  - Access Control Enhanced with Role and Permission Parameters with Authorization Framework Extended with Parameter Engine and Enforcement in SDN Controller.
- **Administration of Access Control in SDN.**
  - SDN-RBACa Administrative Model for Managing roles, Permissions and Network App Authorizations in SDN.
  - Proxy Operations and Custom Permissions for Enhanced Engineering of Administrative Units in SDN.

Access Control Administration in SDN

- App-role and permission-role relations need management.
- In SDN-RBACa administrative model (inspired by Uni-ARBAC):
  - Indirect permission-role assignment.
  - **Permissions** are grouped into permission-pools (tasks).
  - **Tasks**: units of network functions.
  - **Apps** are grouped into app-pools.
  - **Administrative Units** for administering app-role and task-role relations.



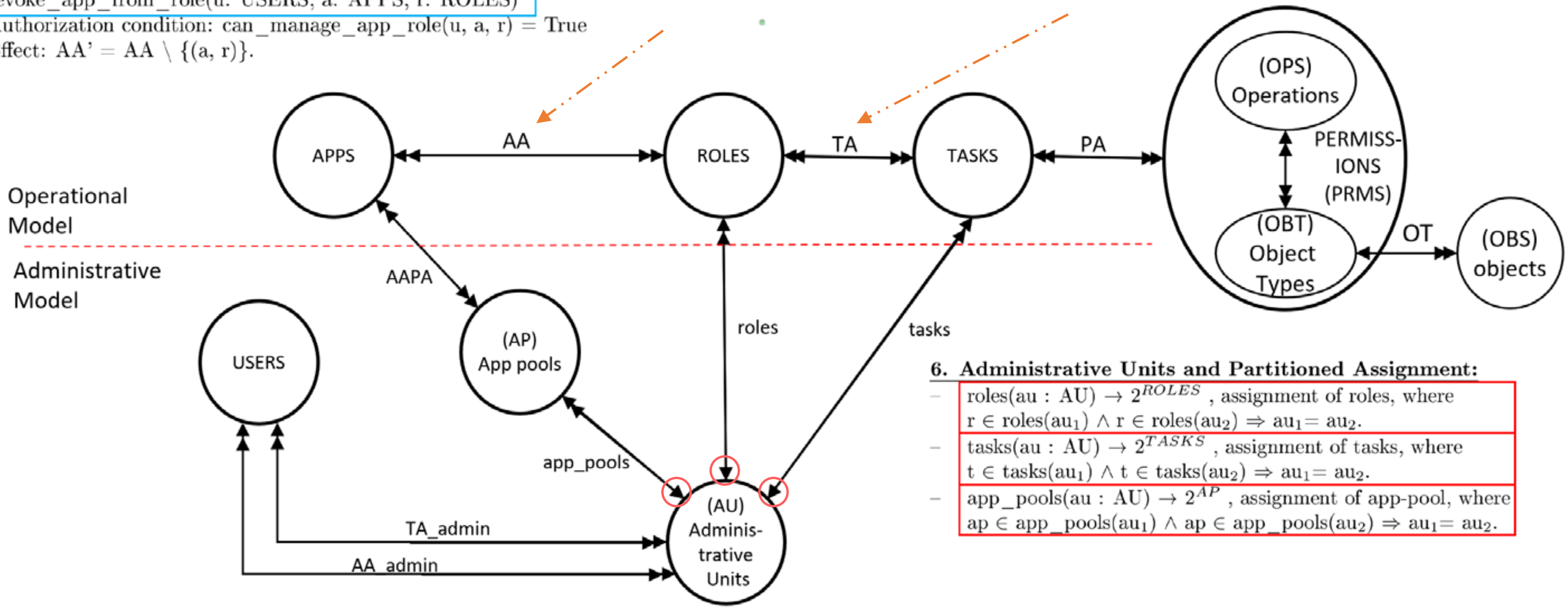
SDN-RBACa Administrative Model

9. **Administrative Actions:**

- assign_task_to_role(u: USERS, t: TASKS, r: ROLES)
  Authorization condition: can_manage_task_role(u, t, r) = True
  Effect: TA' = TA $\cup$ {(t, r)}.
- revoke_task_from_role(u: USERS, t: TASKS, r: ROLES)
  Authorization condition: can_manage_task_role(u, t, r) = True
  Effect: TA' = TA \ {(t, r)}.
- assign_app_to_role(u: USERS, a: APPS, r: ROLES)
  Authorization condition: can_manage_app_role(u, a, r) = True
  Effect: AA' = AA $\cup$ {(a, r)}.
- revoke_app_from_role(u: USERS, a: APPS, r: ROLES)
  Authorization condition: can_manage_app_role(u, a, r) = True
  Effect: AA' = AA \ {(a, r)}.

8. **Administrative User Authorization Functions:**

- can_manage_task_role(u : USERS, t : TASKS, r : ROLES) = $\exists$au$\in$AU : (u , au) $\in$ TA_admin $\wedge$ r $\in$ roles(au) $\wedge$ t $\in$ tasks(au).
- can_manage_app_role(u : USERS, a : APPS, r : ROLES) = $\exists$au$\in$AU : ((u, au) $\in$ AA_admin $\wedge$ r $\in$ roles(au)) $\wedge$ $\exists$ap$\in$AP : ((a , ap) $\in$ AAPA $\wedge$ ap $\in$ app_pools(au)).



6. **Administrative Units and Partitioned Assignment:**

- roles(au : AU) $\to 2^{ROLES}$, assignment of roles, where r $\in$ roles(au_1) $\wedge$ r $\in$ roles(au_2) $\Rightarrow$ au_1= au_2.
- tasks(au : AU) $\to 2^{TASKS}$, assignment of tasks, where t $\in$ tasks(au_1) $\wedge$ t $\in$ tasks(au_2) $\Rightarrow$ au_1= au_2.
- app_pools(au : AU) $\to 2^{AP}$, assignment of app-pool, where ap $\in$ app_pools(au_1) $\wedge$ ap $\in$ app_pools(au_2) $\Rightarrow$ au_1= au_2.

7. **Administrative User Assignment:**

- TA_admin $\subseteq$ USERS $\times$ AU.
- AA_admin $\subseteq$ USERS $\times$ AU.

SDN-RBACa Administrative Model Defenition

1. Basic Sets

- APPS is a finite set of SDN apps.
- OPS is a finite set of operations.
- OBS is a finite set of objects.
- OBTS is a finite set of object types.
- PRMS $\subseteq$ OPS $\times$ OBTS , set of permissions.
- ROLES is a finite set of roles.
- TASKS is a finite set of tasks.
- AP is a finite set of app-pools.
- USERS is a finite set of administrative users.
- AU is a finite set of administrative units.

2. Assignment Relations (operational):

- PA $\subseteq$ PRMS $\times$ TASKS, permission-task assignment relation.
- TA $\subseteq$ TASKS $\times$ ROLES, task-role assignment relation.
- AA $\subseteq$ APPS $\times$ ROLES, app-role assignment relation.
- OT $\subseteq$ OBS $\times$ OBTS, a many-to-one mapping an object to its type, where $(o, t_1) \in \mathrm{OT} \wedge (o, t_2) \in \mathrm{OT} \Rightarrow t_1 = t_2$.

3. Derived Functions (operational):

- type: (o: OBS) $\rightarrow$ OBTS, a function specifying the type of an object. Defined as type(o) $= \{t \in \mathrm{OBTS} \mid (o, t) \in \mathrm{OT}\}$.
- authorized_perms(r: ROLES) $\rightarrow 2^{PRMS}$, defined as authorized_perms$(r) = \{p \in \mathrm{PRMS} \mid \exists t \in \mathrm{TASKS}, \exists r \in \mathrm{ROLES} : (t, r) \in \mathrm{TA} \bigwedge (p, t) \in \mathrm{PA}\}$.

4. App Authorization Function:

- can_exercise_permission(a: APPS, op: OPS, ob: OBS) = $\exists$r $\in$ ROLES : (op, type(ob)) $\in$ authorized_perms(r) $\bigwedge$ (a, r) $\in$ AA.

5. Administrative App-pools Relation:

- AAPA $\subseteq$ APPS $\times$ AP, app to app-pool assignment relation.

6. Administrative Units and Partitioned Assignment:

- roles(au : AU) $\rightarrow 2^{ROLES}$, assignment of roles, where r $\in$ roles(au_1) $\wedge$ r $\in$ roles(au_2) $\Rightarrow$ $au_1 = au_2$.
- tasks(au : AU) $\rightarrow 2^{TASKS}$, assignment of tasks, where t $\in$ tasks(au_1) $\wedge$ t $\in$ tasks(au_2) $\Rightarrow$ $au_1 = au_2$.
- app_pools(au : AU) $\rightarrow 2^{AP}$, assignment of app-pool, where ap $\in$ app_pools(au_1) $\wedge$ ap $\in$ app_pools(au_2) $\Rightarrow$ $au_1 = au_2$.

7. Administrative User Assignment:

- TA_admin $\subseteq$ USERS $\times$ AU.
- AA_admin $\subseteq$ USERS $\times$ AU.

8. Administrative User Authorization Functions:

- can_manage_task_role(u : USERS, t : TASKS, r : ROLES) = $\exists$au$\in$AU : (u , au) $\in$ TA_admin $\bigwedge$ r $\in$ roles(au) $\bigwedge$ t $\in$ tasks(au).
- can_manage_app_role(u : USERS, a : APPS, r : ROLES) = $\exists$au$\in$AU : ((u, au) $\in$ AA_admin $\bigwedge$ r $\in$ roles(au)) $\bigwedge$ $\exists$ap$\in$AP : ((a , ap) $\in$ AAPA $\bigwedge$ ap $\in$ app_pools(au)).

9. Administrative Actions:

- assign_task_to_role(u: USERS, t: TASKS, r: ROLES)
  Authorization condition: can_manage_task_role(u, t, r) = True
  Effect: TA' = TA $\cup$ {(t, r)}.
- revoke_task_from_role(u: USERS, t: TASKS, r: ROLES)
  Authorization condition: can_manage_task_role(u, t, r) = True
  Effect: TA' = TA $\setminus$ {(t, r)}.
- assign_app_to_role(u: USERS, a: APPS, r: ROLES)
  Authorization condition: can_manage_app_role(u, a, r) = True
  Effect: AA' = AA $\cup$ {(a, r)}.
- revoke_app_from_role(u: USERS, a: APPS, r: ROLES)
  Authorization condition: can_manage_app_role(u, a, r) = True
  Effect: AA' = AA $\setminus$ {(a, r)}.

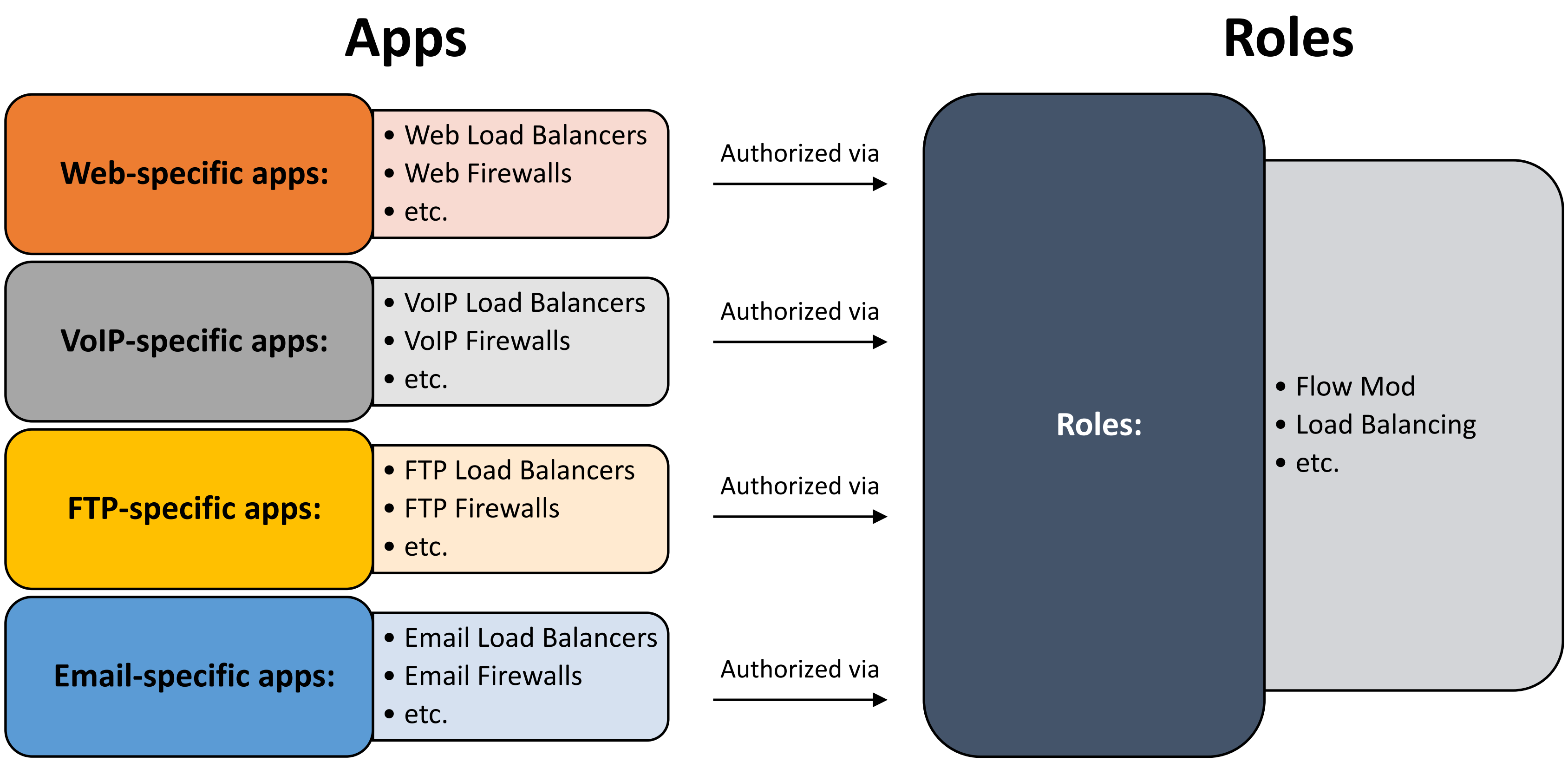
Use Case using SDN-RBACa - Introduction

- In large SDNs, specialized **apps** control/analyze and monitor/inspect specific network **traffic** type**.**
- These apps should be authorized to access only traffic type they handle and not other type (via roles).

Apps			Roles	
Web-specific apps:	• Web Load Balancers • Web Firewalls • etc.	Authorized via →	**Web-specific roles:**	• Web Flow Mod • Web Load Balancing • etc.
VoIP-specific apps:	• VoIP Load Balancers • VoIP Firewalls • etc.	Authorized via →	**VoIP-specific roles:**	• VoIP Flow Mod • VoIP Load Balancing • etc.
FTP-specific apps:	• FTP Load Balancers • FTP Firewalls • etc.	Authorized via →	**FTP-specific roles:**	• Ftp Flow Mod • Ftp Load Balancing • etc.
Email-specific apps:	• Email Load Balancers • Email Firewalls • etc.	Authorized via →	**Email-specific roles:**	• Email Flow Mod • Email Load Balancing • etc.

Use Case using SDN-RBACa - Introduction

- In large SDNs, specialized **apps** control/analyze and monitor/inspect specific network **traffic** type**.**
- These apps should be authorized to access only traffic type they handle and not other type (via roles).



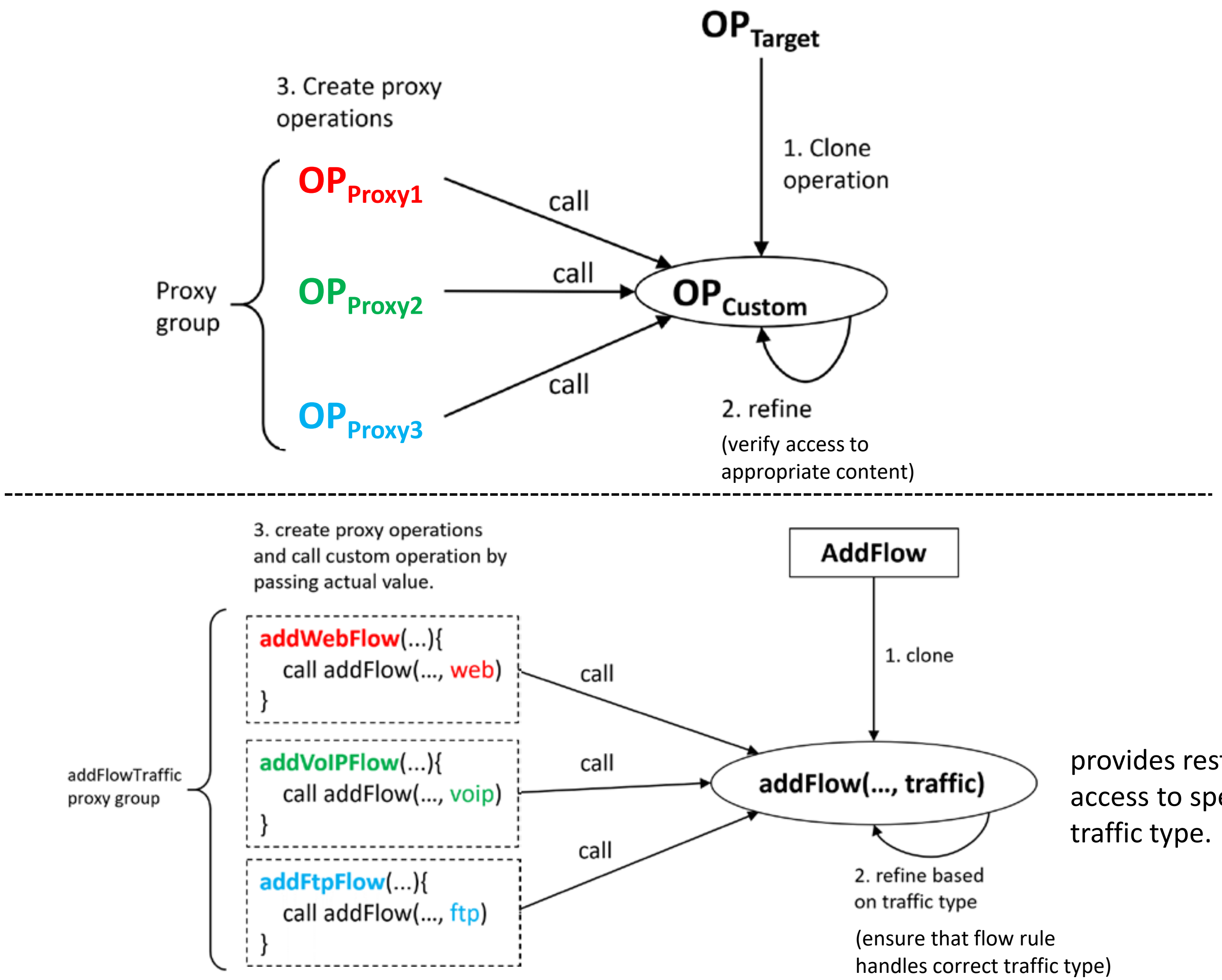
Functional Administrative Units for SDN

- Relations between apps and roles should be managed by different administrative units.

Administrative Units

Admin Unit	Roles / App-Pools
Web Admin Unit	• **Roles**: {Web Flow Mod, Web Load Balancing, etc.} • **App-Pools**: {Web Security, Web Load Balance, etc.}
Email Admin Unit	• **Roles**: {Email Flow Mod, Email Load Balancing, , etc.} • **App-Pools**: {Email Security, Email Load Balance}
VoIP Admin Unit	• **Roles:** {Email Flow Mod, VoIP Load Balancing, etc.}} • **App-Pools:** {VoIP Security, VoIP Load Balance}
FTP Admin Unit	• **Roles**: {FTP Mod Email, FTP Load Balancing, etc.} • **App-Pools**: {FTP Security, FTP Load Balance}

Custom and Proxy Operations



provides restrictive access to specific traffic type.

Custom Permissions

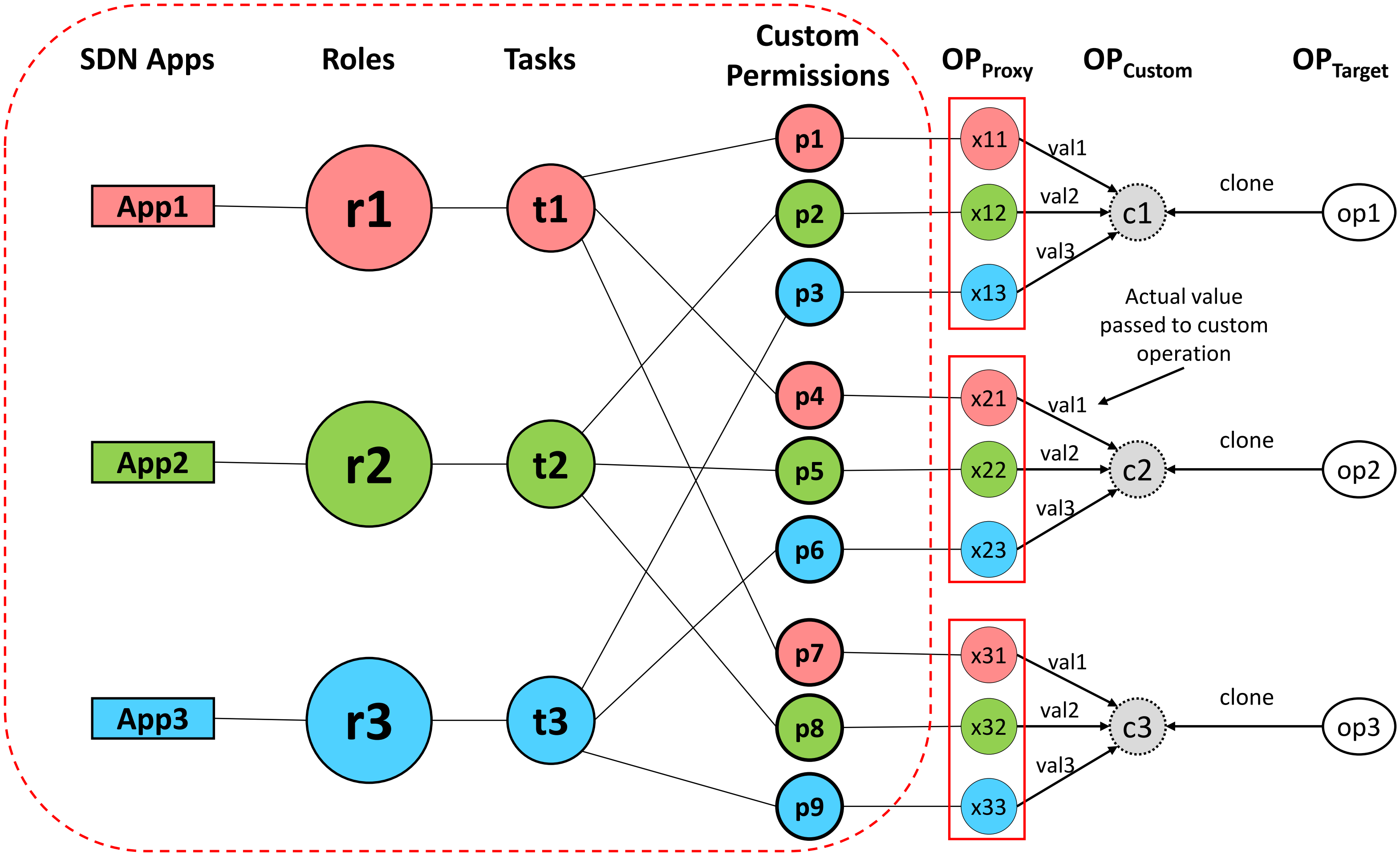
- Custom permissions are those permissions that are created using proxy operations.

(OP_{Proxy_1}, ot)
(OP_{Proxy_2}, ot)
(OP_{Proxy_3}, ot)
…

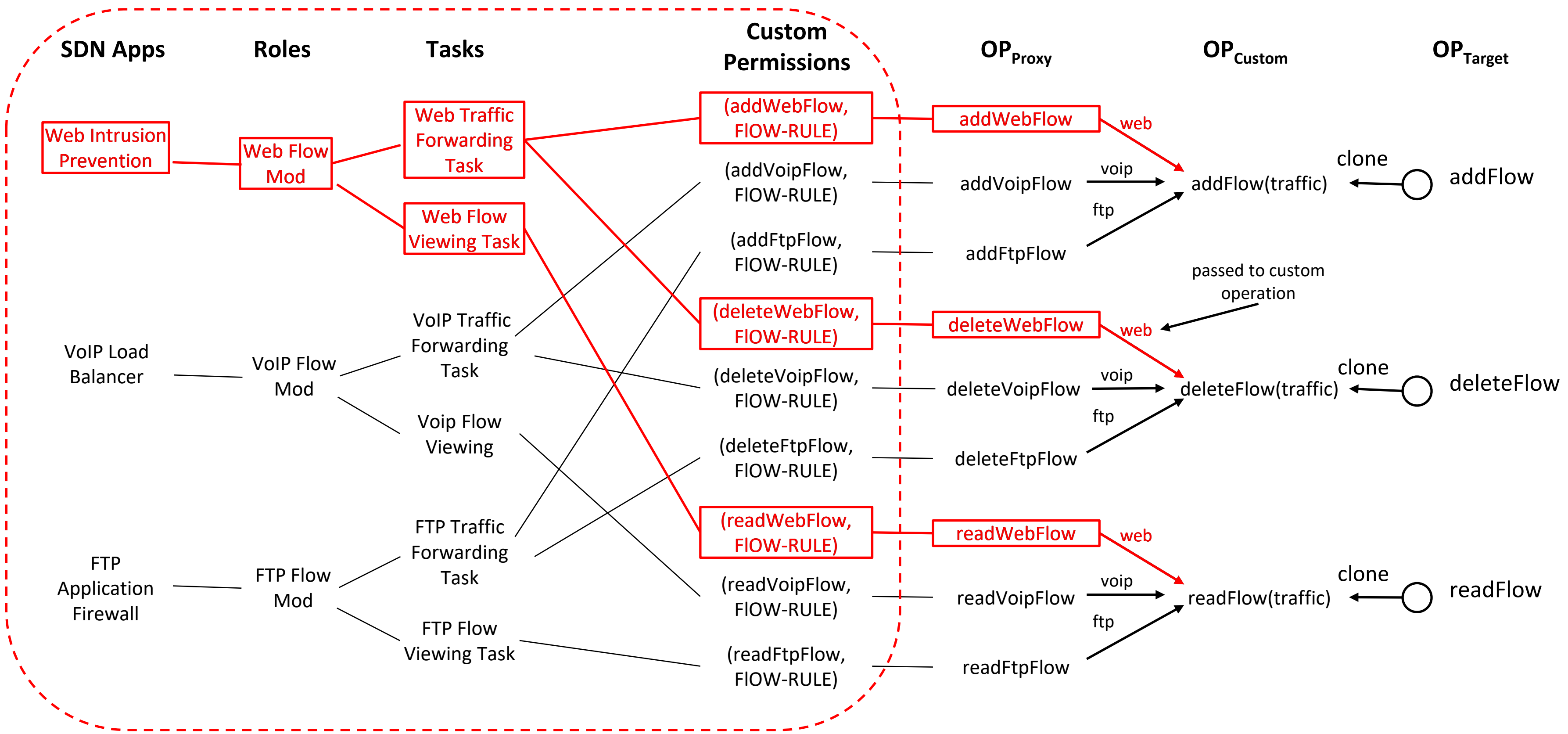
Examples:

(addWebFlow, FLOW-RULE)
(addVoIPFlow, FLOW-RULE)
(addFtpFlow, FLOW-RULE)
(createWebMember, LB-POOL-MEMBER)
(createVoIPMember, LB-POOL-MEMBER)
(createFtpMember, LB-POOL-MEMBER)
(readWebPacketInPayload, PI-PAYLOAD)
(readVoIPPacketInPayload, PI-PAYLOAD)
…

Task and Role Engineering Custom Permissions



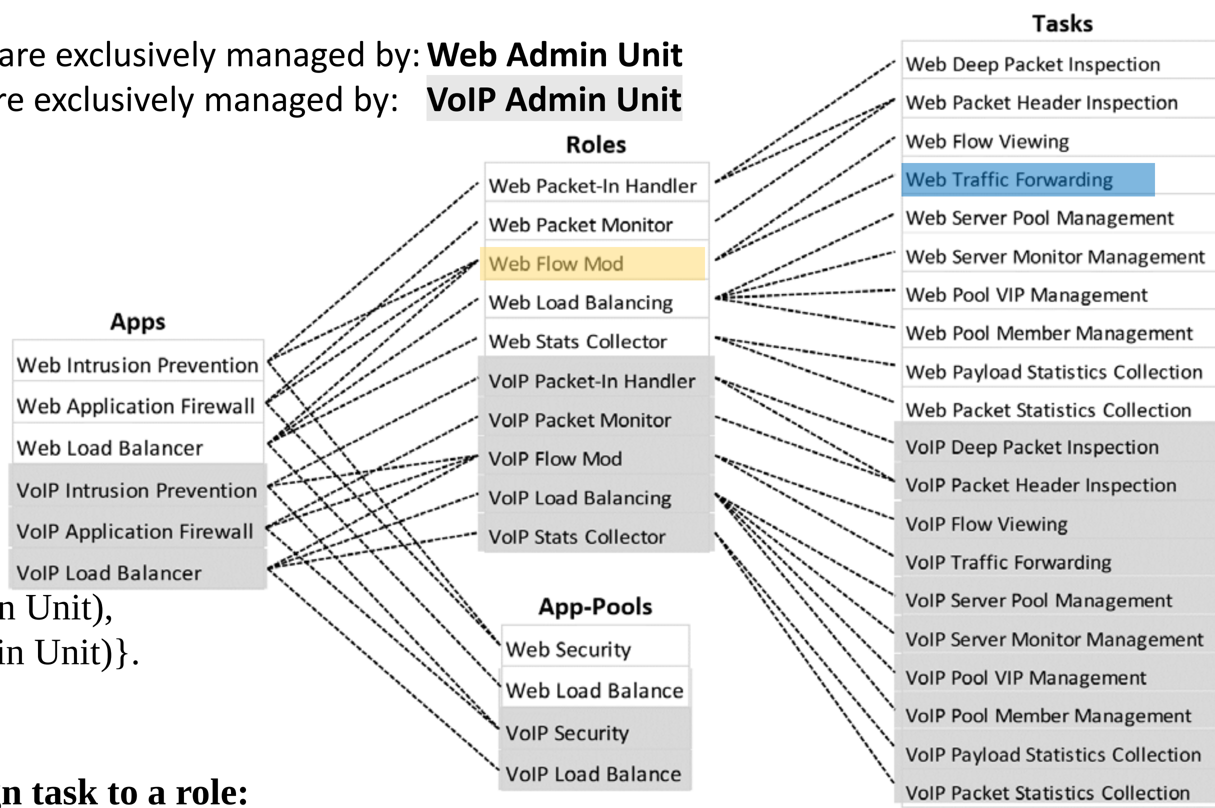
Task and Role Engineering using Custom Permissions - Example



@ Abdullah Al-Alaj

Use-Case and Administrative Actions

Tasks, roles, and app-pools in white are exclusively managed by: **Web Admin Unit**
Tasks, roles, and app-pools in gray are exclusively managed by: **VoIP Admin Unit**

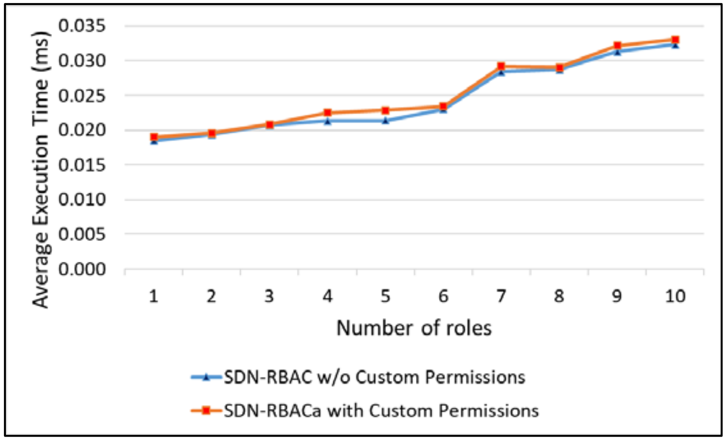


Administrative User Assignment:
TA_admin = {
(web_functions_admin_user, Web Admin Unit),
(voip_functions_admin_user, VoIP Admin Unit)}.

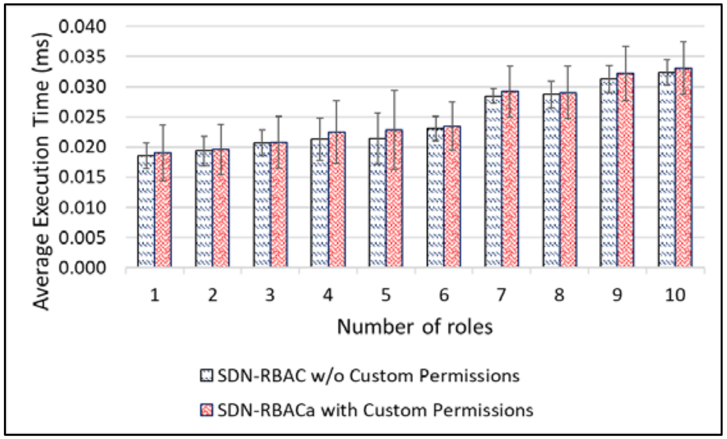
Example:

1. **Administrative Action to assign task to a role:**
assign_task_to_role(web_functions_admin_user, Web Traffic Forwarding Task, Web Flow Mod) **is allowed**.
→**Authorization Function:**
can_manage_task_role(web_functions_admin_user, Web Traffic Forwarding Task, Web Flow Mod) = True.
Reason:
∃Web Admin Unit ∈ AU : ((web_functions_admin_user, Web Admin Unit) ∈ TA_admin) ∧
Web Flow Mod ∈ roles(Web Admin Unit) ∧
Web Traffic Forwarding Task ∈ tasks(Web Admin Unit).

Evaluation and Comparison



- Evaluation of SDN-RBACa operational model with tasks and proxy permissions.
- Test app with 50 proxy operations ops covered by 10 different roles.
- Report authorization time for all 50 requests.
- Different security policies.
- Test repeated 100 times for each security policy.
- Average authorization time is calculated.



- Operational model of SDN-RBACa adds an average of 0.0252 ms overhead on the floodlight controller while SDN-RBAC adds 0.0245 ms on average.

- Using tasks in SDN-RBACa operational model introduces additional variance in the authorization check time.

- The operational model of SDN-RBACa introduces acceptable overhead to the controller for the sake of access control administration.

Conclusion and Future Work

- We presented SDN-RBAC, a model for enabling role based authorization for SDN. SDN-RBAC is implemented and enforced in Floodlight controller.
- We presented ParaSDN, a fine-Grained and Scalable Access Control for SDN Enhanced with Role and Permission Parameters. The Authorization Framework includes Parameter Engine and Enforcement in SDN Controller.
- We presented SDN-RBACa, an administrative model for SND enhanced with Proxy Operations and Custom Permissions.

Future Work:

- Access Control for SDN-Enabled technologies.
- Risk-Aware Access Control for SDN Apps.

Dissertation Publications

Published:

1. Abdullah Al-Alaj, 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.
2. Abdullah Al-Alaj, Ravi Sandhu, and Ram Krishnan. "A Formal Access Control Model for SE-Floodlight Controller." *Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*. ACM, 2019.

Submitted for review:

3. Abdullah Al-Alaj, Ram Krishnan, and Ravi Sandhu. ParaSDN: An Access Control Model for SDN Applications based on Parameterized Roles and Permissions. *In 2020 IEEE 6th International Conference on Collaboration and Internet Computing (CIC).* Atlanta, Georgia, USA, IEEE, 2020.
4. Abdullah Al-Alaj, Ravi Sandhu, and Ram Krishnan. A Model for the Administration of Access Control in Software Defined Networking using Custom Permissions. *In 2020 Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA).* Atlanta, Georgia, USA, IEEE, 2020.

Thank you!

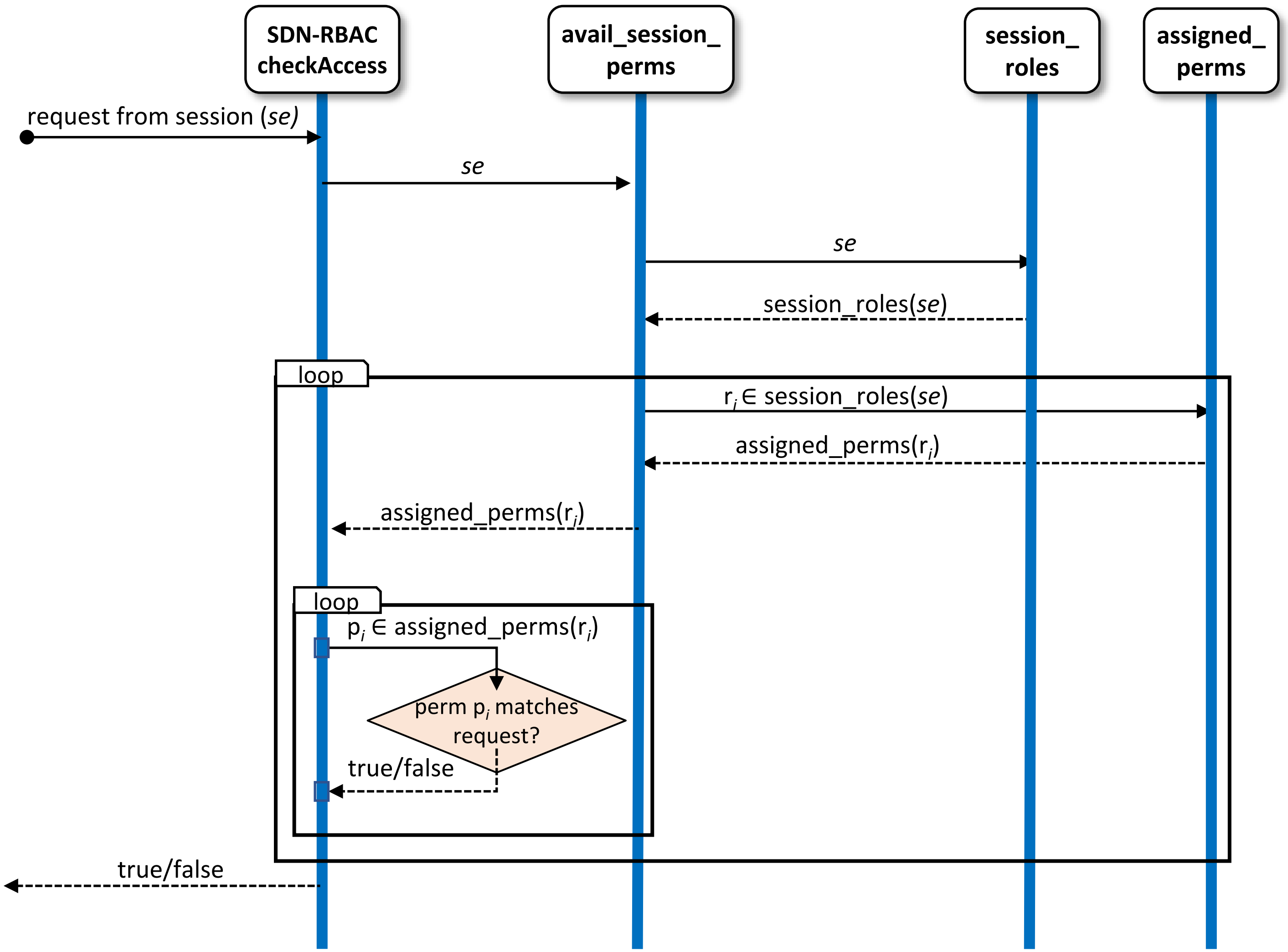
Questions?

Backup Slides

Selected roles in SDN-RBAC

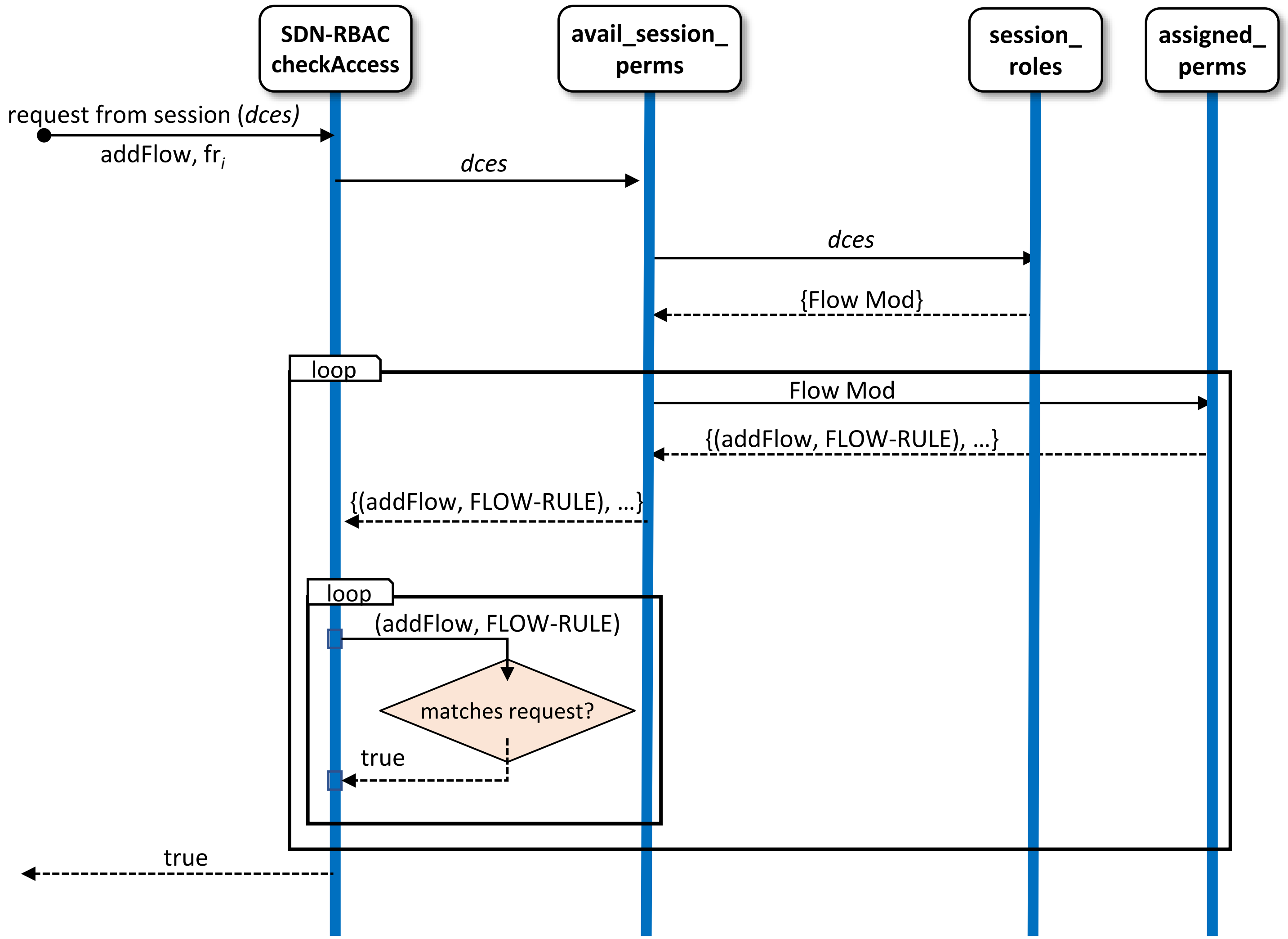
Role	General Functionality
Device Handler	permissions for querying the controller about devices
Bandwidth Monitoring	permissions to read the bandwidth consumption for switch ports.
Flow Mod	permissions to insert/update/delete flow rules into a switch's flow tables.
Link Handler	permissions to get information 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

SDN-RBAC - Check Access



SDN-RBAC App Authorization- Example

- ***dces*** **= DataCapEnforcingSession**



SDN-RBAC: Specifications of System Functions

Session Creation/Deletion

Adding/Dropping Active Role

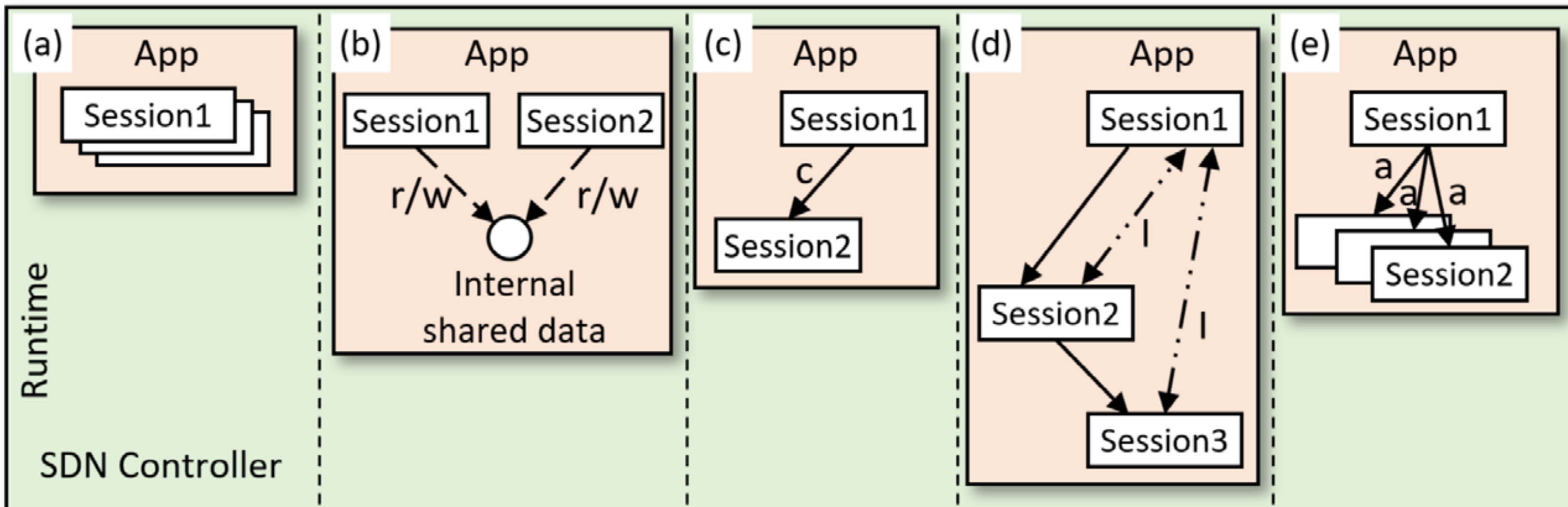
Access Check

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

Apps are authorized based on object type.

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.

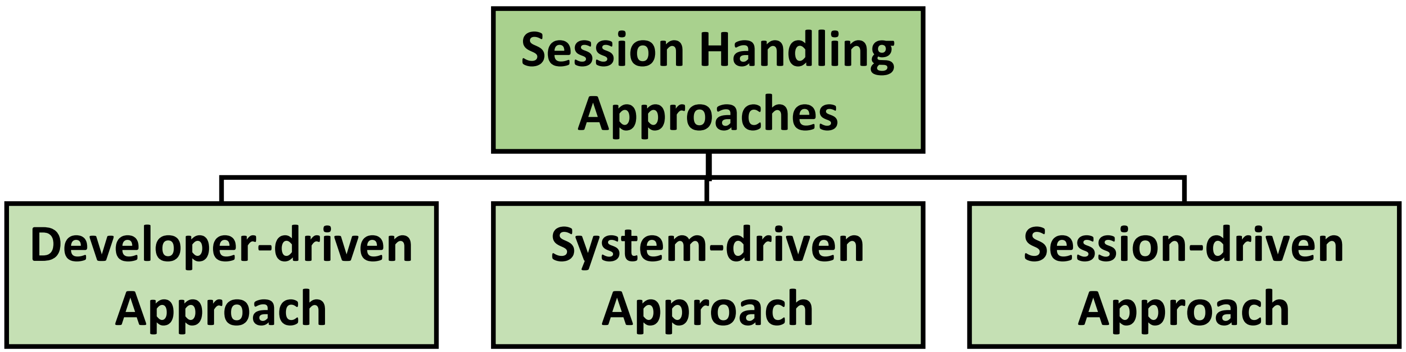
c : condition that triggers session creation.

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

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

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.



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)

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

- Developer has full and prior knowledge of
  - all possible sessions
  - active role set required for each session to achieve its task.
- This information is provided to the controller before app execution.
- The controller knows in advance:
  - what session instances will be created.
  - the tasks that will execute in each session.
  - active role set required for each session.

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

- 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.

Parameter Checking Functions

A. Verifiers:
Language LVerify is used to define each verifier V_i(*s: SESSIONS, op*: OPS, *ob*: OBS, pvpair : PVPAIRS) in VERIFIERS.

B. CandidateVerifiers: a function that maps each object type to its applicable set of verifiers.

CandidateVerifiers(ot: OBTS, pvpairs : $2^{PVPAIRS}$){
    verifiers = {};
    For each $\text{pvpair}_i \in$ pvpairs **do**
        V_i = **param_verifier**(ot, pvpair_i.par);
        verifiers := verifiers $\cup$ {($V_i \times \text{pvpair}_i$)};
    return verifiers;
}

C. ParamCheck: a function that checks an object against all candidate verifiers until the first failure is discovered or a true is returned as the final outcome.

ParamCheck(s: SESSIONS, op: OPS, ob: OBS, pvpairs: $2^{PVPAIRS}$){
    For each ($V_i \times \text{pvpair}_i$) $\in$ **CandidateVerifiers**(type(ob), pvpairs) **do**
        if $\neg V_i$(s, op, ob, pvpair_i)
            return false;
    return true;
}

App Authorization Function

Function	Authorization Condition
checkAccess(s: SESSIONS, op: OPS, ob: OBS)	$\exists$pr $\in$ PROLES : pr $\in$ session_roles(s), $\exists$pp $\in$ PPRMS : (pp, pr) $\in$ PPA $\wedge$ (op, type(ob)) = (pp.op, pp.ot) $\wedge$ **ParamCheck**(s, op, ob, pp.PVPAIRS) = True.

Verifier Example

A.3. VRuleSwitch(s: SESSIONS, op: OPS, ob: OBS, pvpair : PVPAIRS){

//assume a request from app Data Usage Cap Mngr via DataCapEnforcingSession with the following:

//ob = flow_rule$_{[switch_id=0x2, tcp_dst=80, ...]}$

//pvpair = (dept, {CS})

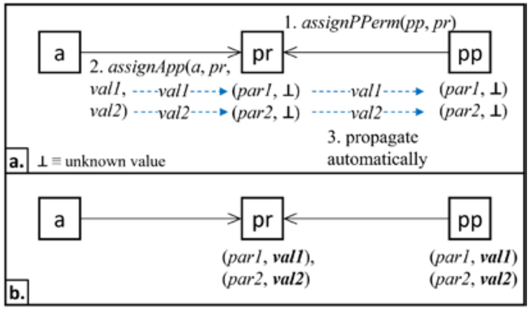
//switches(CS) = {0x1, 0x2}

($\exists$d $\in$ pvpair.val : ob.switch_id $\in$ switches(d)); //will return true

}

Parameter Value Assignment

- Parameter values, assigned via assignApp administrative action, propagate automatically from role parameters to permission parameters.



Example:

	(Flow Mod, {(dept, ⊥), (traffic, ⊥)})	((addFlow, FLOW-RULE), {(dept, ⊥), (traffic, ⊥)})	
assignPPerm(pp_i, pr_i)	**pr_i =** (Flow Mod, { **(dept, ⊥),** **(traffic, ⊥)** })	**pp_i =** ((addFlow, FLOW-RULE), { **(dept, ⊥),** **(traffic, ⊥)** })	**1. Parameterized permission assigned to parameterized role. Parameter values are unknown.**
assignApp(**a** = Data Usage Cap Mngr, **pr_i,** **{CS},** **web**)	**pr_i =** (Flow Mod, { **(dept, {CS}),** **(traffic, web)** })	**pp_i =** ((addFlow, FLOW-RULE), { **(dept, ⊥),** **(traffic, ⊥)** })	**2. Parameter value assigned to parameterized role via assignApp administrative action.**
assignApp(**a** = Data Usage Cap Mngr, **pr_i,** **{CS},** **web**)	**pr_i =** (Flow Mod, { **(dept, {CS}),** **(traffic, web)** })	**pp_i =** ((addFlow, FLOW-RULE), { **(dept, {CS}),** **(traffic, web)** })	**3. Parameter values propagate from parameterized role to parameterized permission.**
	(Flow Mod, {(dept, {CS}), (traffic, web)})	((addFlow, FLOW-RULE), {(dept, {CS}), (traffic, web)})	

ParaSDN : App Authorization Example

- checkAccess(DataCapEnforcingSession, addFlow, fow_rule$_{[switch_id=0x2,tcp_dst=80,...]}$) ≡

  ∃(**Flow Mod**, {(dept, {CS}), (traffc, web)}) ∈ PROLES :

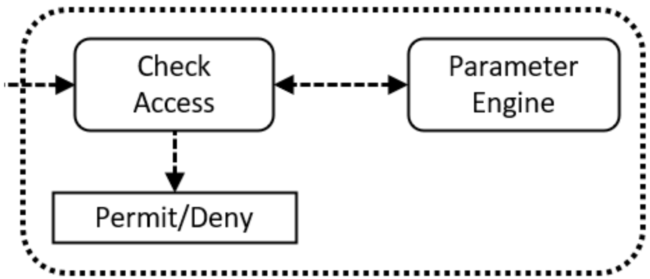
  (**Flow Mod**, {(dept, {CS}), (traffic, web)}) ∈ session_roles(DataCapEnforcingSession), = true

  ∃((**addFlow, FLOW-RULE)**, {(dept, {CS}), (traffic, web)}) ∈ PPRMS :

  (((**addFlow, FLOW-RULE)**, {(dept, {CS}), (traffic, web)}), (**Flow Mod**, {(dept, {CS}), (traffic, web)})) ∈ PPA ∧ = true

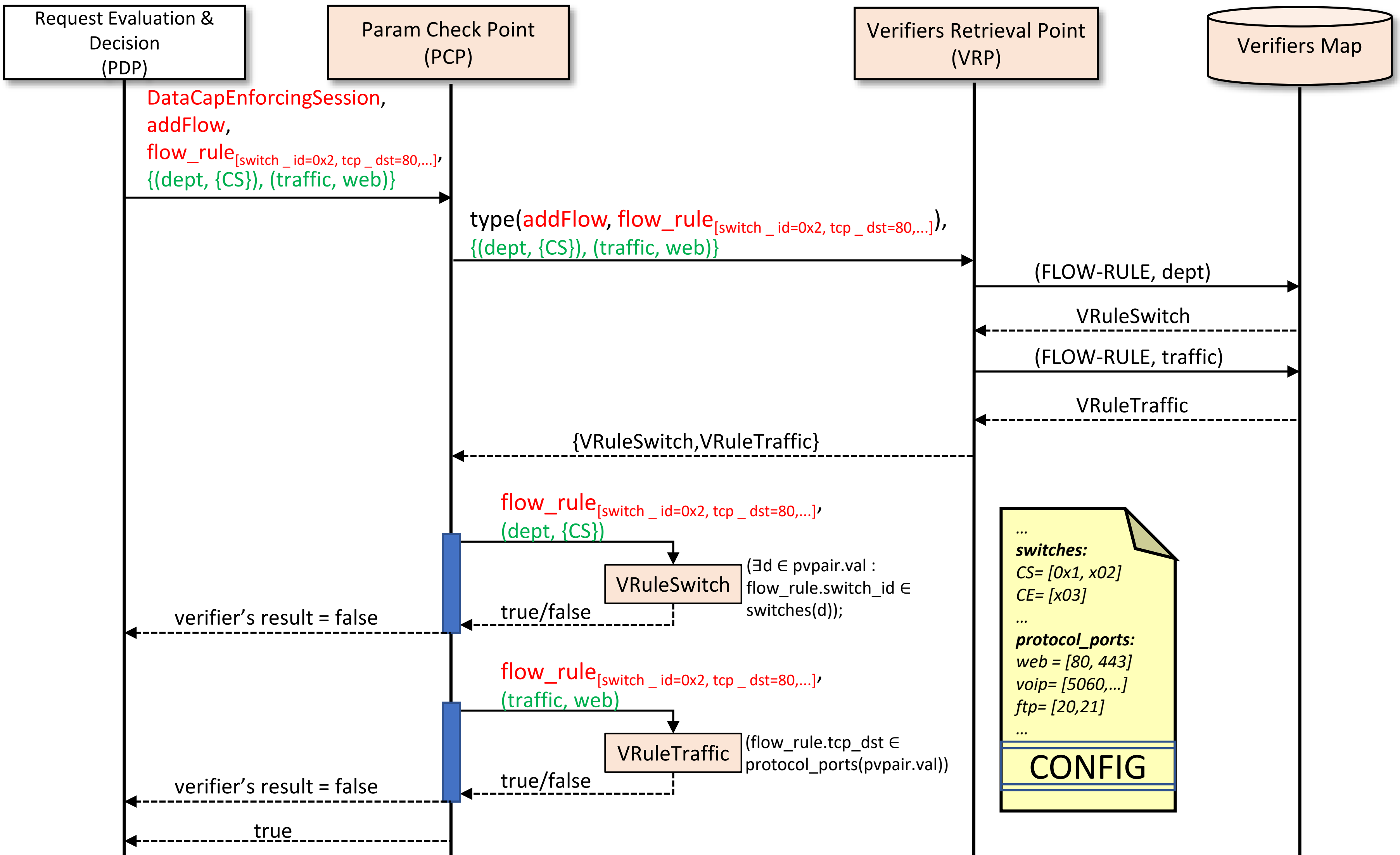
  (addFlow, type(flow_rule$_{[switch_id=0x2,\ tcp_dst=80,...]}$)) = (addFlow, FLOW-RULE) ∧ = true

  ParamCheck(DataCapEnforcingSession, addFlow, flow_rule$_{[switch_id=0x2,\ tcp_dst=80,...]}$, {(dept, {CS}), (traffic, web)}) = True.



PPA = {
…,
(((addFlow, FLOW-RULE), {(dept, {CS}), (traffic, web)}),
(Flow Mod, {(dept, {CS}), (traffic, web)})),
…
}.

Parameter Engine - Use Case Example



Administrative Actions - Examples

1 . Examples of Authorization Functions:
– can_manage_task_role(web_functions_admin_user, Web Traffic Forwarding Task, Web Flow Mod) = True ***Reason***: ∃Web Admin Unit ∈ AU : ((web_functions_admin_user, Web Admin Unit) ∈ TA_admin) ⋀ Web Flow Mod ∈ roles(Web Admin Unit) ⋀ Web Traffic Forwarding Task ∈ tasks(Web Admin Unit).
– can_manage_task_role(voip_functions_admin_user, Web Server Pool Management Task, Web Load Balancing) = False ***Reason***: Web Load Balancing ∈ roles(Web Admin Unit) ⋀ Web Server Pool Management Task ∈ tasks(Web Admin Unit), however, (voip_functions_admin_user, Web Admin Unit) ∉ TA_admin.
– can_manage_app_role(web_apps_admin_user, Web Intrusion Prevention App, Web Flow Mod) = True ***Reason***: ∃Web Admin Unit ∈ AU : ((web_apps_admin_user, Web Admin Unit) ∈ AA_admin) ⋀ Web Flow Mod ∈ roles(Web Admin Unit)] ⋀ ∃Web Security Pool ∈ AP: (Web Intrusion Prevention App, Web Security Pool) ∈AAPA ⋀ Web Security Pool ∈ app_pools(Web Admin Unit).
– can_manage_app_role(web_apps_admin_user, VoIP Application Firewall App, VoIP Flow Mod) = False ***Reason***: VoIP Flow Mod ∈ roles(VoIP Admin Unit) ⋀ (VoIP Application Firewall App , VoIP Security) ∈ AAPA ⋀ VoIP Security ∈ app_pools(VoIP Admin Unit)], however, (web_apps_admin_user, VoIP Admin Unit) ∉ AA_admin.
2 . Examples of Administrative Actions:
– assign_task_to_role(web_functions_admin_user, Web Traffic Forwarding Task, Web Flow Mod) is allowed ***Reason***: can_manage_task_role(web_functions_admin_user, Web Traffic Forwarding Task, Web Flow Mod) = True
– revoke_task_from_role(voip_functions_admin_user, Web Server Pool Management Task, Web Load Balancing) is not allowed ***Reason***: can_manage_task_role(voip_functions_admin_user, Web Server Pool Management Task, Web Load Balancing) = False
– assign_app_to_role(web_apps_admin_user, Web Intrusion Prevention App, Web Flow Mod) is allowed ***Reason***: can_manage_app_role(web_apps_admin_user, Web Intrusion Prevention App, Web Flow Mod) = True
– revoke_app_from_role(web_apps_admin_user, VoIP Application Firewall App, VoIP Flow Mod) is not allowed ***Reason***: can_manage_app_role(web_apps_admin_user, VoIP Application Firewall App, VoIP Flow Mod) = False

Administrative User Assignment:
– TA_admin = {(web_functions_admin_user, Web Admin Unit), (voip_functions_admin_user, VoIP Admin Unit)}.
– AA_admin = {(web_apps_admin_user, Web Admin Unit), (voip_apps_admin_user, VoIP Admin Unit)}.

Configuration of the SDN-RBACa in a use case (1)

1.Basic Sets

- APPS = {Web Intrusion Prevention App, Web Application Firewall App, Web Load Balancer App}.
- OPS = {
  readWebPacketInPayload, readWebPacketHeader, readWebFlow, addWebFlow,
  updateWebFlow, deleteWebFlow, createWebPool, listWebPools, removeWebPool,
  updateWebPool, createWebMonitor, listWebMonitors, removeWebMonitor, updateWebMonitor,
  createWebVip, listWebVips, removeWebVip, updateWebVip, createWebMember,
  listWebMembersByPool, removeWebMember, updateWebMember, readWebFlowByteCount,
  readAggWebFlowByteCount, readWebFlowPacketCount, readAggWebFlowPacketCount
  }.
- OBS = set of all objects of types PI-PAYLOAD, PI-HEADER, FLOW-RULE, LB-POOL, LB-MONITOR, LB-VIP, LB-POOL-MEMBER, and FLOW-STATS.
- OBTS = {PI-PAYLOAD, PI-HEADER, FLOW-RULE, LB-POOL, LB-MONITOR, LB-VIP, LB-POOL-MEMBER, FLOW-STATS}.
- PRMS = {
  (readWebPacketInPayload, PI-PAYLOAD), (readWebPacketHeader, PI-HEADER),
  (readWebFlow, FLOW-RULE), (addWebFlow, FLOW-RULE), (updateWebFlow, FLOW-RULE),
  (deleteWebFlow, FLOW-RULE), (createWebPool, LB-POOL), (listWebPools, LB-POOL),
  (removeWebPool, LB-POOL), (updateWebPool, LB-POOL), (createWebMonitor, LB-MONITOR),
  (listWebMonitors, LB-MONITOR), (removeWebMonitor, LB-MONITOR),
  (updateWebMonitor, LB-MONITOR), (createWebVip, LB-VIP), (listWebVips, LB-VIP),
  (removeWebVip, LB-VIP), (updateWebVip, LB-VIP), (createWebMember, LB-POOL-MEMBER),
  (listWebMembersByPool, LB-POOL-MEMBER), (removeWebMember, LB-POOL-MEMBER),
  (updateWebMember, LB-POOL-MEMBER), (readWebFlowByteCount, FLOW-STATS),
  (readAggWebFlowByteCount, FLOW-STATS), (readWebFlowPacketCount, FLOW-STATS),
  (readAggWebFlowPacketCount, FLOW-STATS)
  }.
- ROLES = {Web Packet-In Handler, Web Packet Monitor, Web Flow Mod, Web Load Balancing, Web Stats Collector}.
- TASKS = {
  Web Deep Packet Inspection Task, Web Packet Header Inspection Task,
  Web Flow Viewing Task, Web Traffic Forwarding Task, Web Server Pool Management Task,
  Web Server Monitor Management Task, Web Pool VIP Management Task,
  Web Pool Member Management Task, Web Payload Statistics Collection Task,
  Web Packet Statistics Collection Task
  }.
- AP = {Web Load Balance Pool, Web Security Pool}.
- USERS = {web_functions_admin_user, web_apps_admin_user}.
- AU = {Web Admin Unit}.

Configuration of the SDN-RBACa in a use case (2)

2. Assignment Relations (operational):

- PA = {
{(readWebPacketInPayload, PI-PAYLOAD), (readWebPacketHeader, PI-HEADER),
(readWebFlow, FLOW-RULE)} × {Web Deep Packet Inspection Task} ∪
{(readWebPacketHeader, PI-HEADER), (readWebFlow, FLOW-RULE)} ×
{Web Packet Header Inspection Task} ∪ {(readWebFlow, FLOW-RULE)} × {Web Flow Viewing Task} ∪
{(addWebFlow, FLOW-RULE), (updateWebFlow, FLOW-RULE), (deleteWebFlow, FLOW-RULE)} ×
{Web Traffic Forwarding Task} ∪
{(createWebPool, LB-POOL), (listWebPools, LB-POOL), (removeWebPool, LB-POOL),
(updateWebPool, LB-POOL)} × {Web Server Pool Management Task} ∪
{(createWebMonitor, LB-MONITOR), (listWebMonitors, LB-MONITOR),
(removeWebMonitor, LB-MONITOR), (updateWebMonitor, LB-MONITOR)} ×
{Web Server Monitor Management Task} ∪ {(createWebVip, LB-VIP), (listWebVips, LB-VIP),
(removeWebVip, LB-VIP), (updateWebVip, LB-VIP)} × {Web Pool VIP Management Task} ∪
{(createWebMember, LB-POOL-MEMBER), (listWebMembersByPool, LB-POOL-MEMBER),
(removeWebMember, LB-POOL-MEMBER), (updateWebMember, LB-POOL-MEMBER)} ×
{Web Pool Member Management Task} ∪ {(readWebFlowByteCount, FLOW-STATS),
(readAggWebFlowByteCount, FLOW-STATS)} × {Web Payload Statistics Collection Task} ∪
{(readWebFlowPacketCount, FLOW-STATS), (readAggWebFlowPacketCount, FLOW-STATS)} ×
{Web Packet Statistics Collection Task}}.
- TA = {{Web Deep Packet Inspection Task, Web Packet Header Inspection Task} × {Web Packet-In Handler} ∪
{Web Packet Header Inspection Task} × {Web Packet Monitor} ∪
{Web Flow Viewing Task, Web Traffic Forwarding Task} × {Web Flow Mod} ∪
{Web Server Pool Management Task, Web Server Monitor Management Task, Web Pool VIP Management Task,
Web Pool Member Management Task} × {Web Load Balancing} ∪
{Web Payload Statistics Collection Task, Web Packet Statistics Collection Task} × {Web Stats Collector}}.
- AA = {{Web Intrusion Prevention App} × {Web Packet-In Handler, Web Flow Mod} ∪
{Web Application Firewall App} × {Web Packet Monitor, Web Flow Mod} ∪
{Web Load Balancer App} × {Web Flow Mod, Web Load Balancing, Web Stats Collector}}.
- OT = {(all payloads in packet-in message, PI-PAYLOAD), (all packet header objects, PI-HEADER),
(all flow-rules, FLOW-RULE), (all server pools, LB-POOL), (all server monitors, LB-MONITOR),
(all pools virtual IPs, LB-VIP), (all pool members, LB-POOL-MEMBER),
(all flow statistics in flow rules, FLOW-STATS)}.

3. Administrative App-pools Relation:

- AAPA = {
(Web Intrusion Prevention App, Web Security Pool),
(Web Application Firewall App, Web Security Pool),
(Web Load Balancer App, Web Load Balance Pool)}.

4. Administrative Units and Partitioned Assignment:

- roles(Web Admin Unit) = {Web Packet-In Handler, Web Packet Monitor, Web Flow Mod,
Web Load Balancing, Web Stats Collector}.
- tasks(Web Admin Unit) = {Web Deep Packet Inspection Task, Web Packet Header Inspection Task,
Web Flow Viewing Task, Web Traffic Forwarding Task, Web Server Pool Management Task,
Web Server Monitor Management Task, Web Pool VIP Management Task, Web Pool Member Management Task,
Web Payload Statistics Collection Task, Web Packet Statistics Collection Task}.
- app_pools(Web Admin Unit) = {Web Load Balance Pool, Web Security Pool}.

5. Administrative User Assignment:

- TA_admin = {(web_functions_admin_user, Web Admin Unit)}.
- AA_admin = {(web_apps_admin_user, Web Admin Unit)}.

Access Request Authorization Check with Proxy Operation

```
The method net.floodlightcontroller.staticflowentry.IStaticFlowEntryPusherService.addWebFlow
is called by session net.floodlightcontroller.webtestapp.WebTestAppSession
01:34:14.691 WARN [n.f.rbac.RBAC:Thread-12] SDN-RBAC: security violation, "Access denied".
Unauthorized access requested by session (WebTestAppSession)
Reason: MatchField:TCP_DST - Incorrect port (25) in flow rule
Active roles set for this session: [Web Flow Mod]
01:34:14.824 INFO [n.f.l.i.LinkDiscoveryManager:Scheduled-3] Sending LLDP packets out of all
```