Information Models of Interface to Network Security Functions (I2NSF)
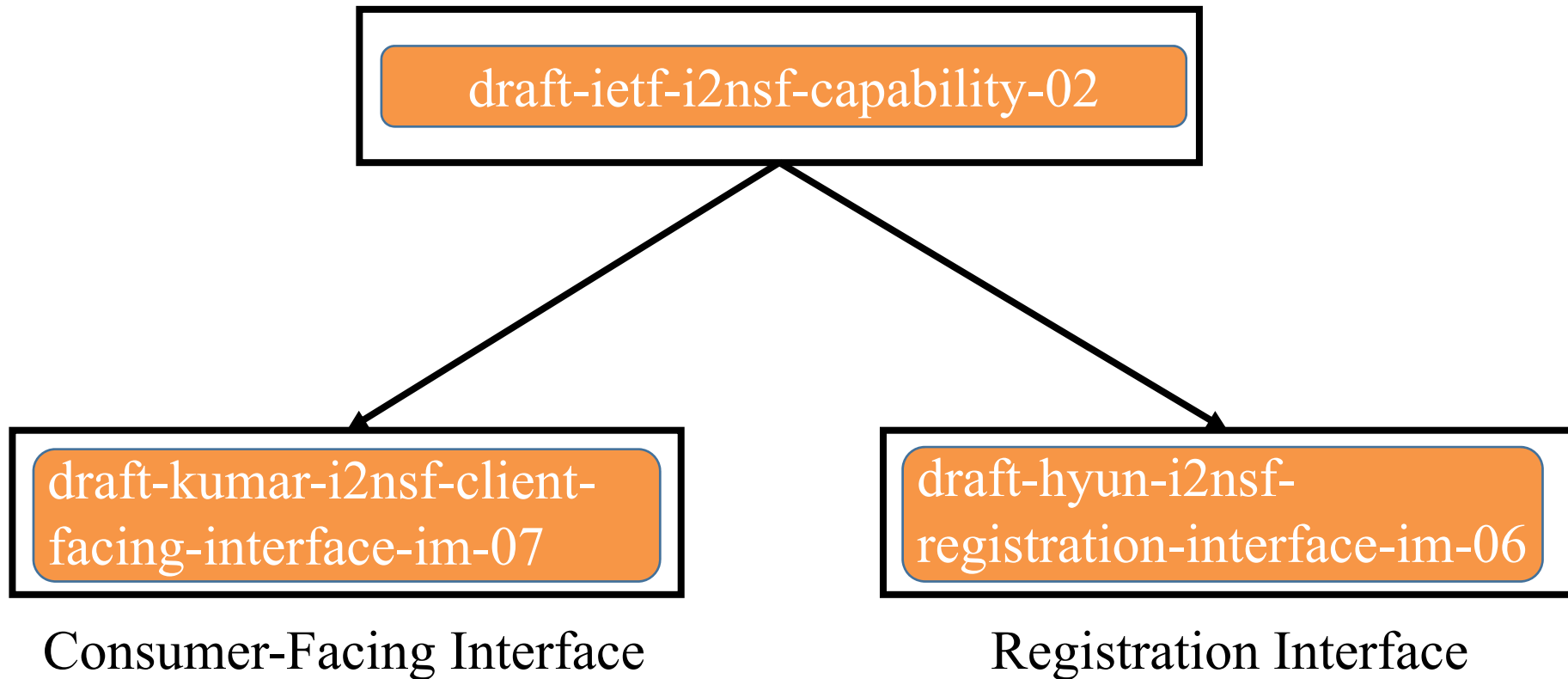
IETF 102, Montreal
July 18, 2018

Seungjin Lee [Presenter]

Information Models of I2NSF

- draft-kumar-i2nsf-client-facing-interface-im-07
  – Client-Facing Interface (i.e., Consumer-Facing Interface)

- draft-hyun-i2nsf-registration-interface-im-06
  – Registration Interface

Information Models of I2NSF

Information Model for Consumer-Facing Interface to Security Controller

(draft-kumar-i2nsf-client-facing-interface-im-07)

R. Kumar, A. Lohiya, D. Qi,

N. Bitar, S. Palislamovic, L. Xia and J. Jeong

Updates from the Previous Versions

- The following changes are made from:
  - draft-kumar-i2nsf-client-facing-interface-im-05
  - draft-kumar-i2nsf-client-facing-interface-im-06

- RBAC (Role-Based Access Control) section is added for synchronizing the requirement of CFI IM.

- The description of condition clauses consisting the condition object is added.

- The diagrams representing high-level abstraction of CFI, condition-clause, RBAC and the objects consisting the IM.

Updates of Version
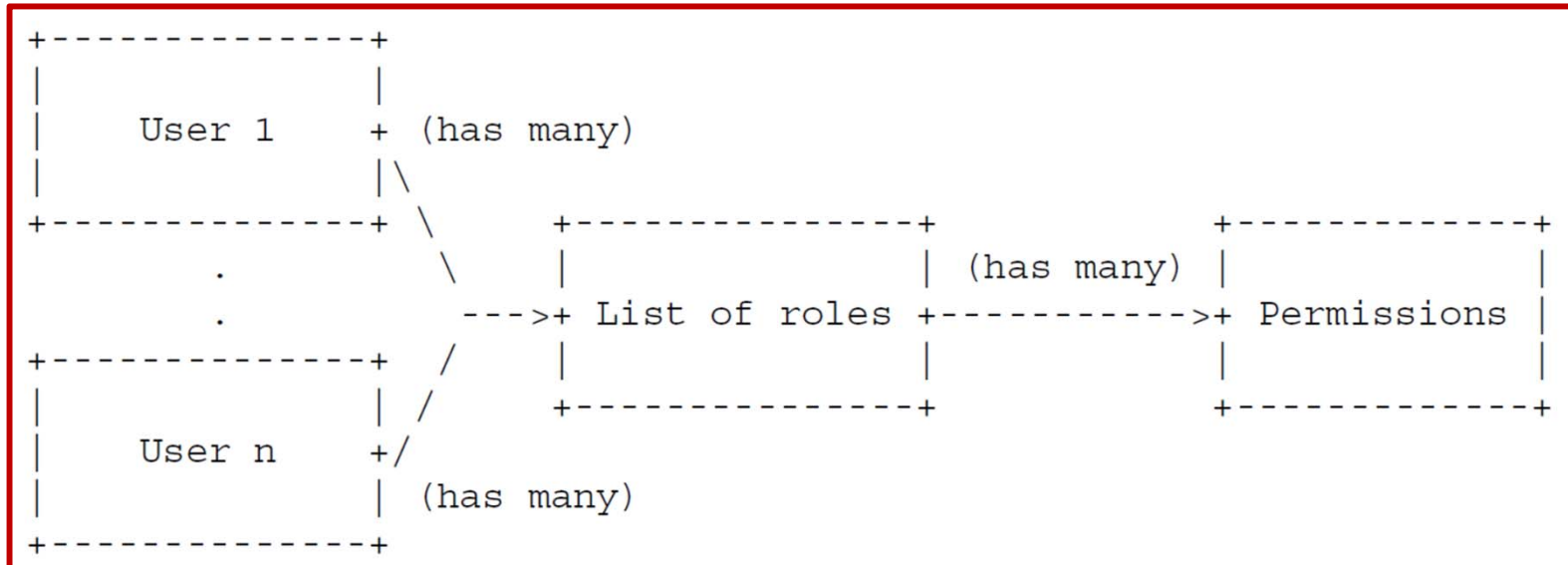
Information Model for Consumer-Facing-Interface

- Diagram for a High-level Abstraction of Consumer-Facing Interface
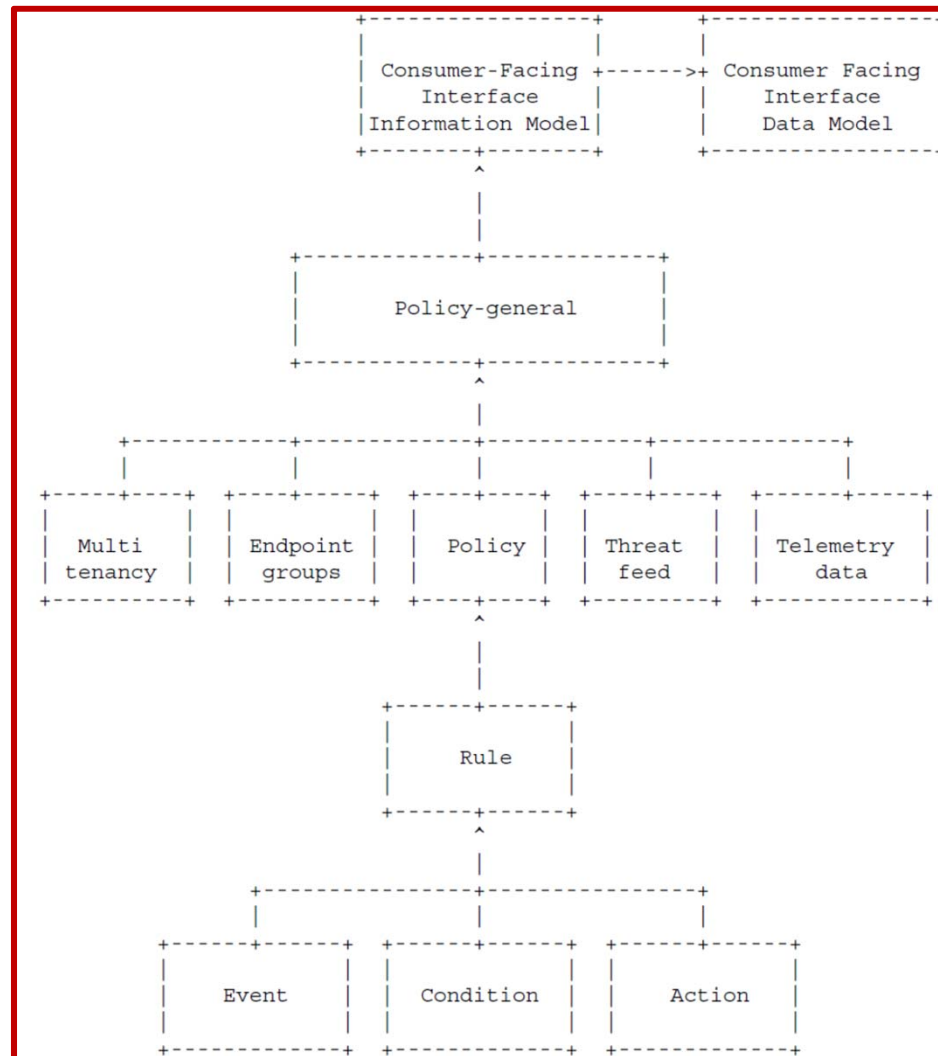
Updates of Version

Information Model for Consumer-Facing-Interface

- Role-Based Access Control (RBAC) provides a powerful and centralized control within a network.

- The required steps to build RBAC are described.

```
+--------------+
|              |
|   User 1     + (has many)
|              |\
+--------------+ \         +----------------+                  +------------+
       .          \        |                | (has many)  |            |
       .           --->+ List of roles +------------>+ Permissions |
+--------------+  /      |                |                  |            |
|              | /       +----------------+                  +------------+
|   User n     +/
|              | (has many)
+--------------+
```

Updates of Version

Information Model for Consumer-Facing-Interface

- The description of condition clauses consisting the condition object is added.
- The condition object is made of condition clauses that consists of three tuples; variable, operator and value.

```
                                                            1..n
                                                    +---------------+
                                                    |               |
                                +------------->+  Policy rule   |
                                |                   |               |
                1..n            |                   +---------------+
            +--------+--------+
            |                 |
   +----------+Condition clause +----------+
   |        |                 |          |
   |        +--------+--------+          |
   |                 ^                   |
   |                 |                   |
   1..n  |          1..n    |          1..n    |
+-------+-------+ +-------+-------+ +-------+-------+
|             | |             | |             |
|   Variable  | |   Operator  | |    Value    |
|             | |             | |             |
+--------------+ +--------------+ +--------------+
```
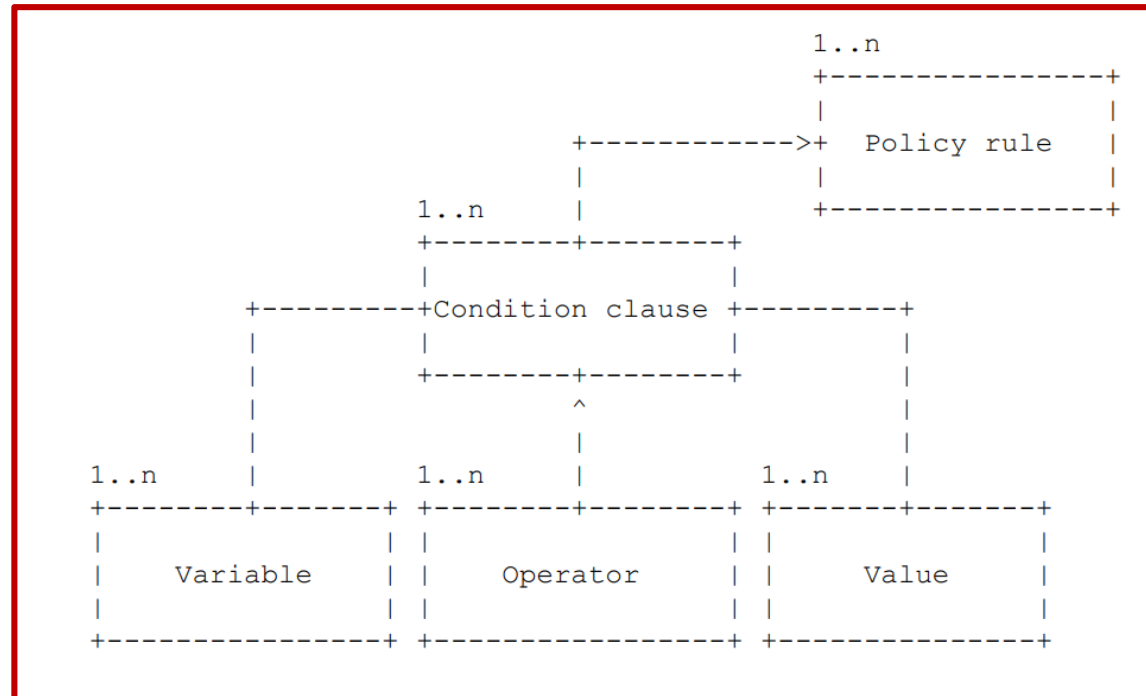
Next Steps

- **WG Adoption Call** in IETF 102

- We will investigate the expansion of the description for each object with popular security services (e.g., firewall, web-filter and DDoS-attack mitigation).

- Also, we will apply the semantics used in a condition clause to the Event sub-model and Action sub-model.

- Conjunction of condition and event for action

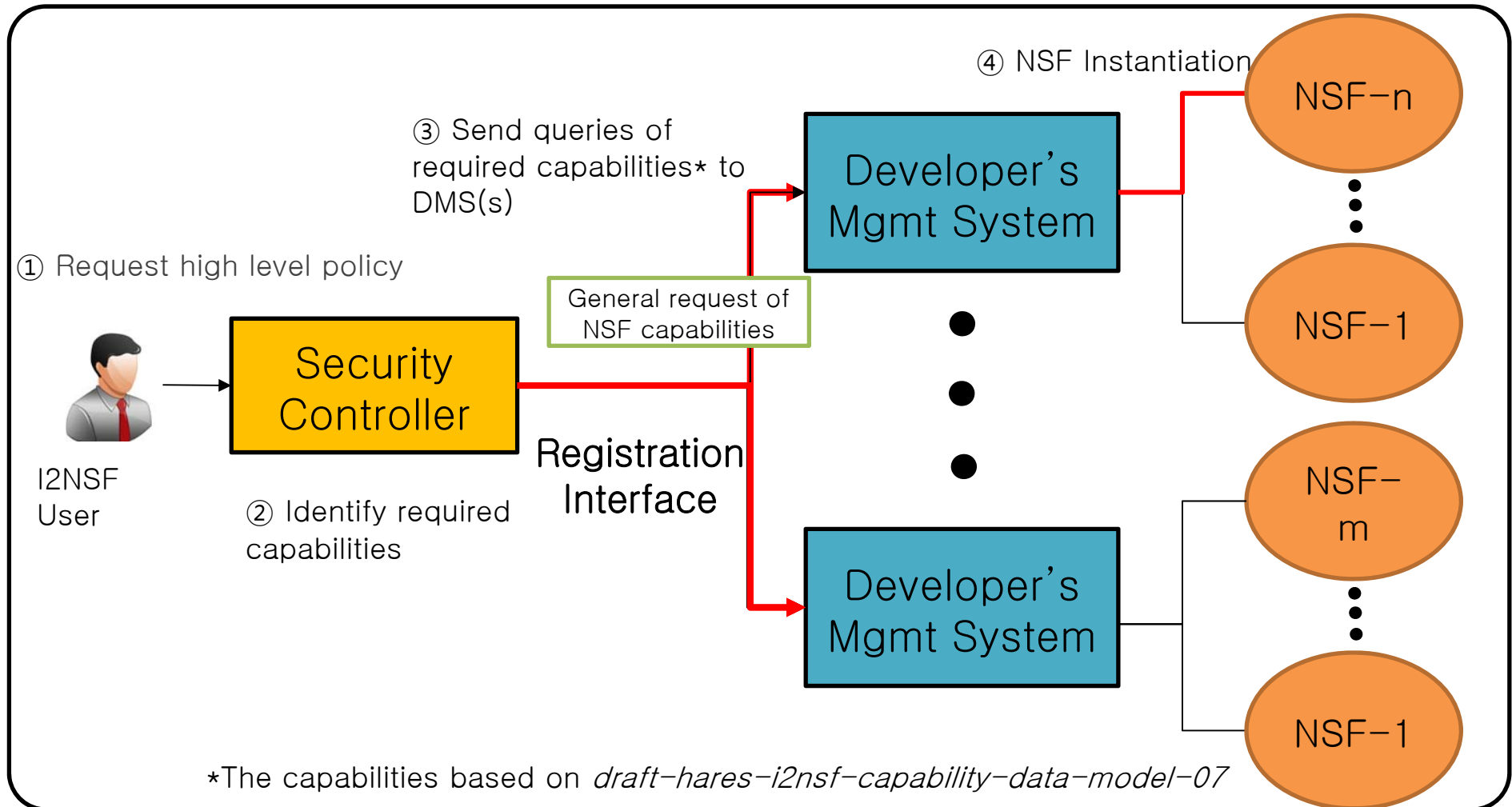# Registration Interface Information Model
(draft-hyun-i2nsf-registration-interface-im-06)

Jaehoon (Paul) Jeong, Sangwon Hyun,

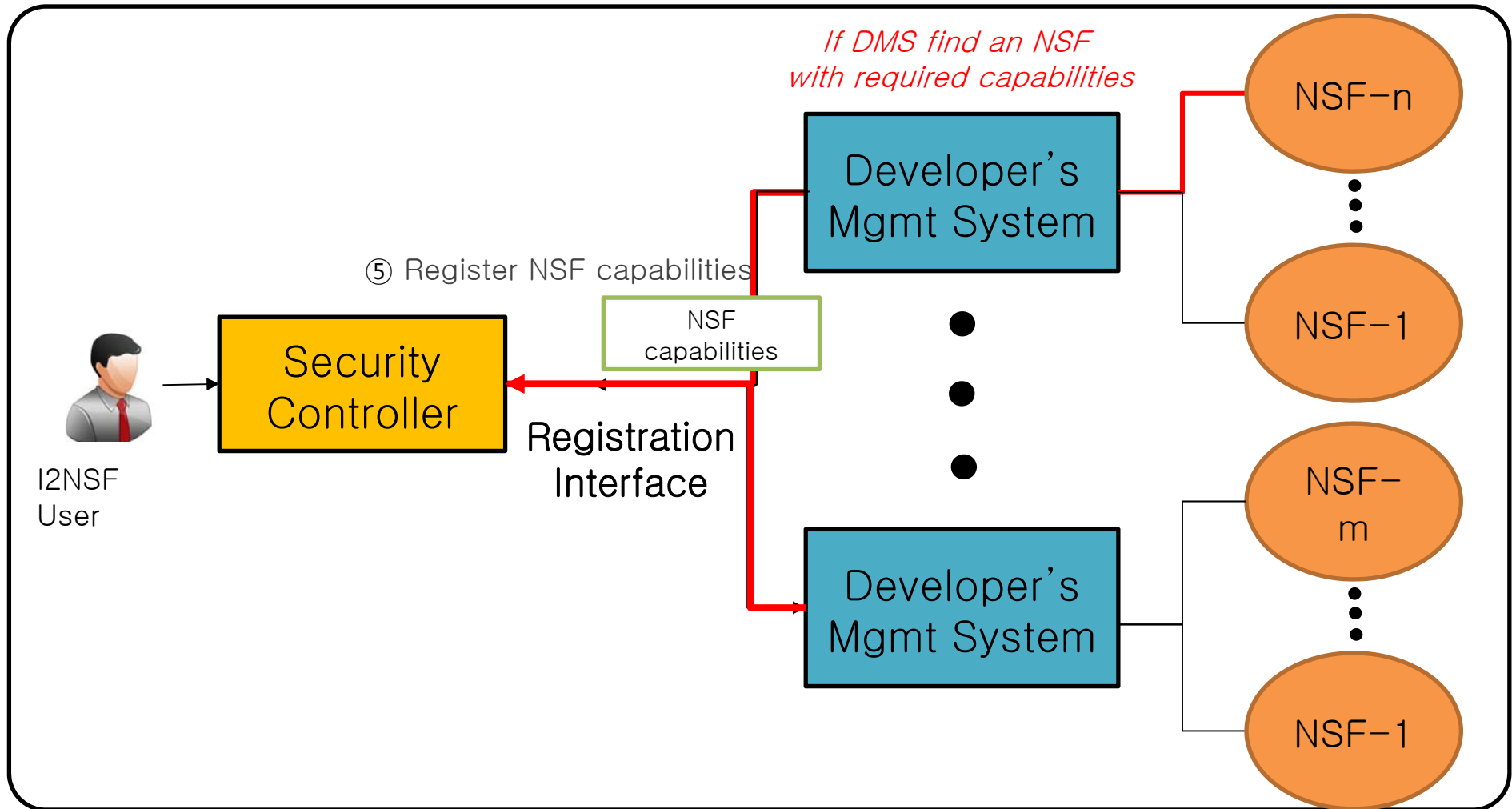Taekyun Roh, Sarang Wi and Jungsoo Park

Updates from the Previous Versions

- The following changes are made from:
  - draft-hyun-i2nsf-registration-interface-im-05
  - draft-hyun-i2nsf-registration-interface-im-06

- Section 4 has been revised to discuss about <u>updating an existing NSF instance via Registration Interface</u>.

- Figures 2, 3 and 4 have been updated according to the above change.

- Appendix A is clarified to discuss <u>the use of the registration interface related to lifecycle management</u>.

- The references have been updated to reflect the latest documents.

Capability-based NSF Search (1/2)

④ NSF Instantiation

③ Send queries of required capabilities* to DMS(s)

① Request high level policy

NSF-n

Developer's Mgmt System

General request of NSF capabilities

I2NSF User

Security Controller

NSF-1

Registration Interface

② Identify required capabilities

NSF-m

Developer's Mgmt System

NSF-1

*The capabilities based on *draft-hares-i2nsf-capability-data-model-07*

Capability-based NSF Search (2/2)

If DMS find an NSF with required capabilities

⑤ Register NSF capabilities

NSF capabilities

Security Controller

I2NSF User

Registration Interface

Developer's Mgmt System

Developer's Mgmt System

NSF-n

NSF-1

NSF-m

NSF-1

Updating Existing NSF Instances

① Capability of NSF added or changed

NSF-n

Developer's Mgmt System

NSF-1

Registration Interface

Security Controller

② Update NSF capabilities

Developer's Mgmt System

NSF-m

NSF-1
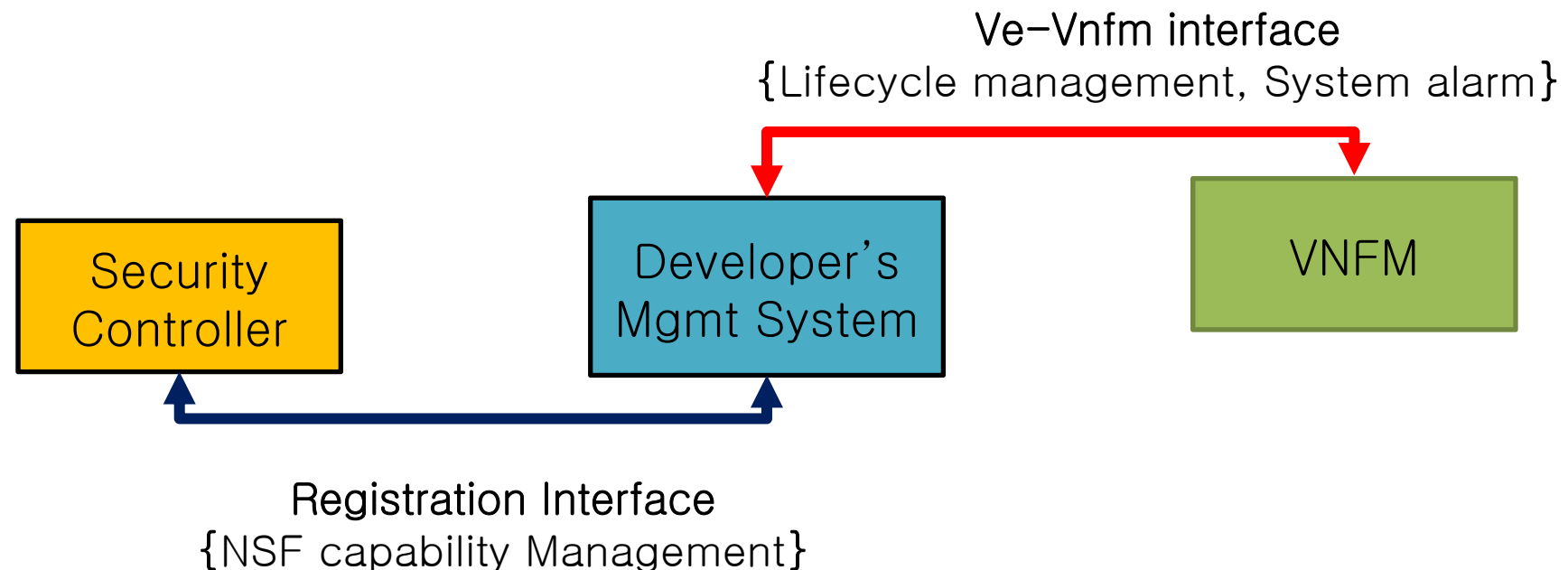
I2NSF User

*The capabilities based on *draft-hares-i2nsf-capability-data-model-07*
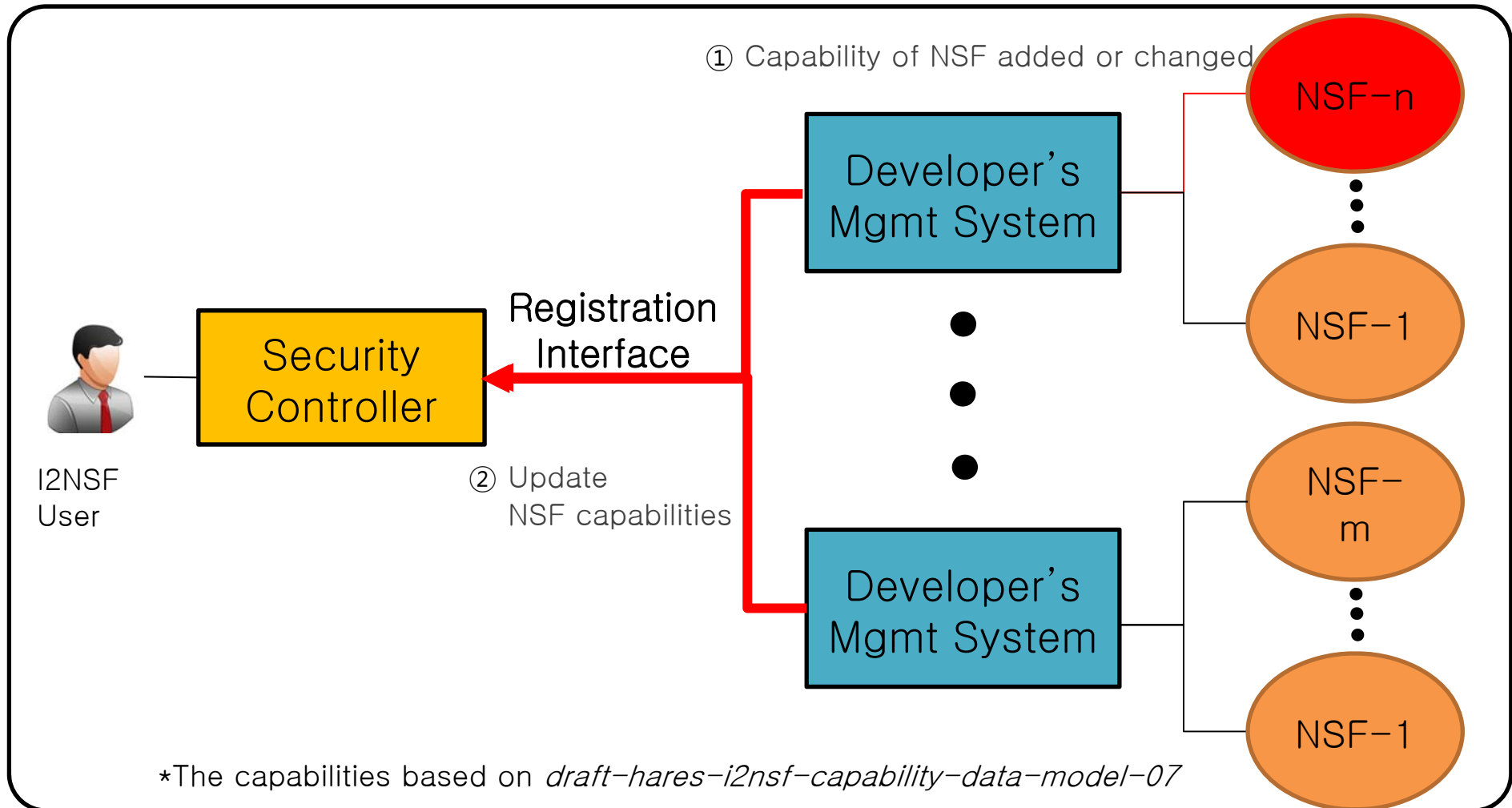
Lifecycle Management Mechanism

- NFV Interface called Ve-Vnfm is used to
  - Request lifecycle management of NSF (VNF) instances
  - Provide VNFM with
    - Some configuration/state information of NSF instances that may be useful for lifecycle management.
    - Some information (e.g., system alarm) obtained via Monitoring interface.

Ve-Vnfm interface
{Lifecycle management, System alarm}

| Security Controller | Developer's Mgmt System | VNFM |

Registration Interface
{NSF capability Management}

Next Steps

- WG Adoption Calls for Two IM Drafts
  - draft-kumar-i2nsf-client-facing-interface-im-07
  - draft-hyun-i2nsf-registration-interface-im-06

- We will implement the following functions in the IETF-103 Hackathon
  - Instantiation/de-instantiation/updating functions through Registration Interface in OpenStack and Open Source MANO (OSM) environment