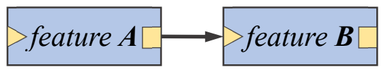
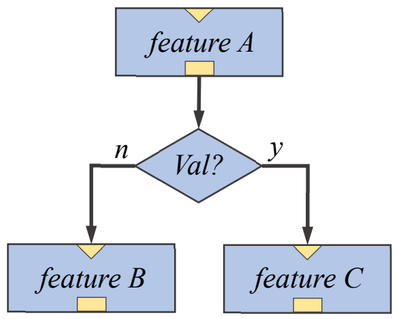
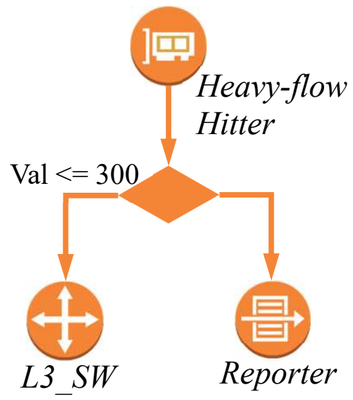
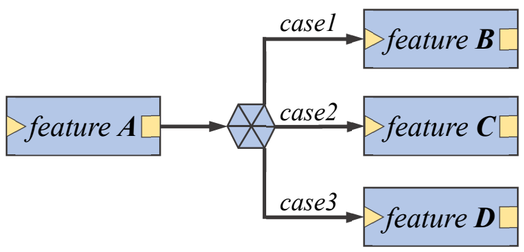
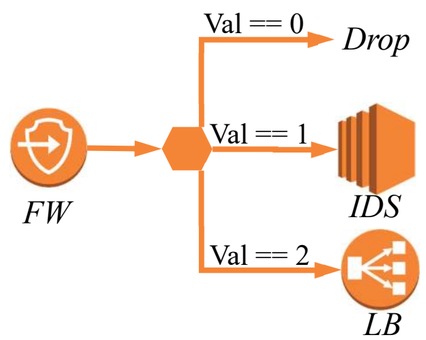
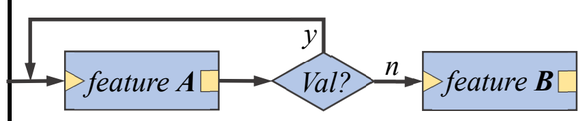
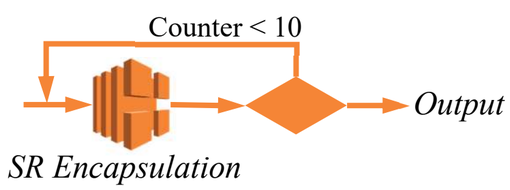
Type	Control Flow Chart	Syntax	Use Case
Sequence		**BNF:** sequence_declaration ::= *feature_name* => *feature_name* **Note:** ✓ *feature_name* is the name of features on the device. **Example:** *A => B*	 **Command:** *L2_SW => FW* **Description:** The feature of *Firewall* follows the feature of *L2_SW*. Sequence structure is the most widely adopted logic.
Selection	if-else: 	**BNF:** If_else_declaration ::= *feature_name* => (*feature_name* : *feature_name* ? validator_declaration) validator_declaration ::= metadata_declaration op *constant* metadata_declaration ::= *feature_name.global_metadata* op ::= > | >= | == | <= | < | != **Note:** ✓ The left value of the validator is the global metadata field assigned for each packet passing among features. **Example:** *A => (C : B ? A.global_metadata == 1)*	 **Command:** *HITTER => (L3_SW : REPORTER ? HEHITTER.global_metadata <= 300)* **Description:** The feature of *Heavy-flow Hitter* identifies the heavy flow and sends the flow to a *Reporter*, otherwise to the feature of *L3_SW*.
	multibranch: 	**BNF:** multibranch_declaration ::= *feature_name* => ({case_declaration :} case_declaration # metadata_declaration) case_declaration ::= *const* @ *feature_name* **Note:** ✓ The *const* number in case_declaration can be set dynamically at runtime. **Example:** *A => (n1@B : n2@C : n3@D # A.global_metadata)*	 **Command:** *FW => (0@DROP:1@IDS:2@LB # FW.global_metadata)* **Description:** The feature of *Firewall* drops the packet (block); sends the packet to an *IDS* (alert); sends the packet to a *Load Balancer* (pass).
Loop		**BNF:** loop_declaration ::= *feature_name* => (*self* : *feature_name* ? validator_declaration) validator_declaration ::= metadata_declaration op *constant* metadata_declaration ::= *feature_name.global_counter* op ::= > | >= | == | <= | < | != **Note:** The orchestrator compiler guarantees the number of loops is determinate. The *constant* number can be set at runtime. **Example:** *A => (self : B ?A.global_counter > 0)*	 **Command:** *SEGMENT_ROUTING_ENCAP => (SEGMENT_ROUTING_ENCAP : Output ? SEG_ROUTNG_ENCAP.global_counter < 10)* **Description:** By the loop structure, operators can readily implement the feature with one time of encapsulation, and dynamically set the counter for each packet to precisely control the number of times for header encapsulation. The *SR Encapsulation* will execute encapsulation for 10 times, then output the packet.