Open Flow 1.3.1 Support: Controller View

Anilkumar Vishnoi, Abhijit Kumbhare

IBM

Controller's view of openflow switch:

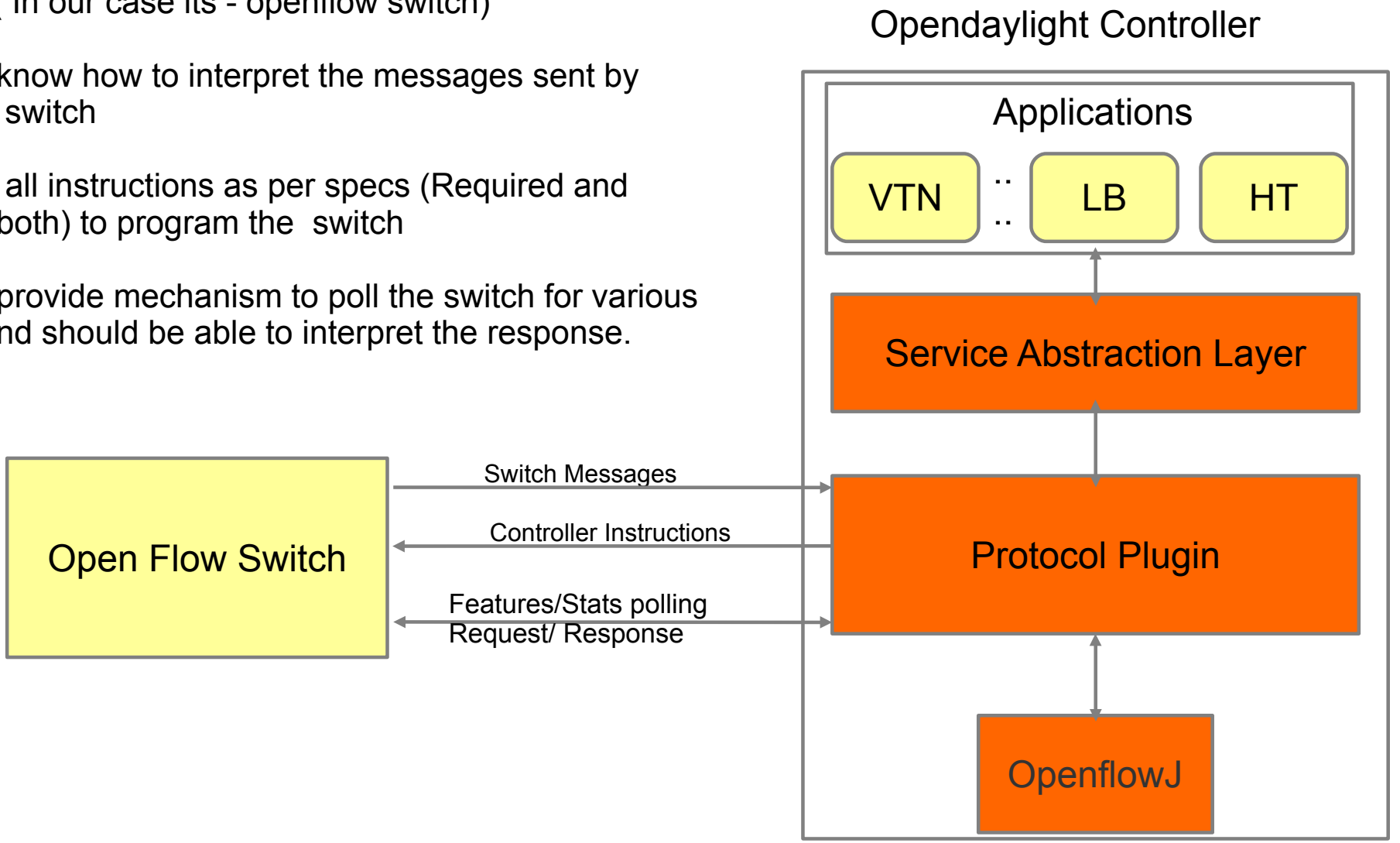
Controller's responsibilities :

• Provide mechanism to connect and interact with underlying platform (In our case its - openflow switch)

• Should know how to interpret the messages sent by openflow switch

• Provide all instructions as per specs (Required and Optional both) to program the switch

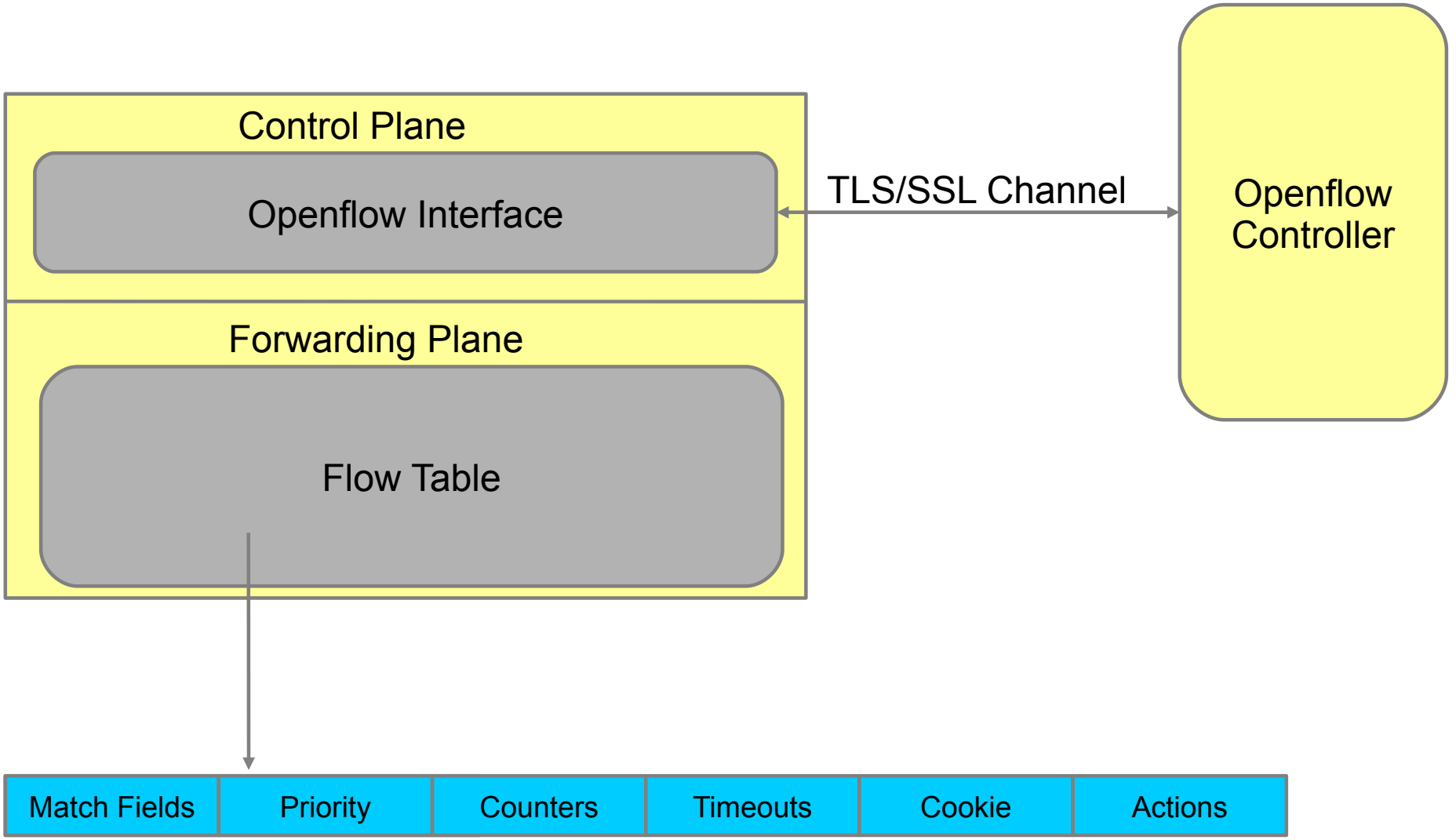
• Should provide mechanism to poll the switch for various details, and should be able to interpret the response.



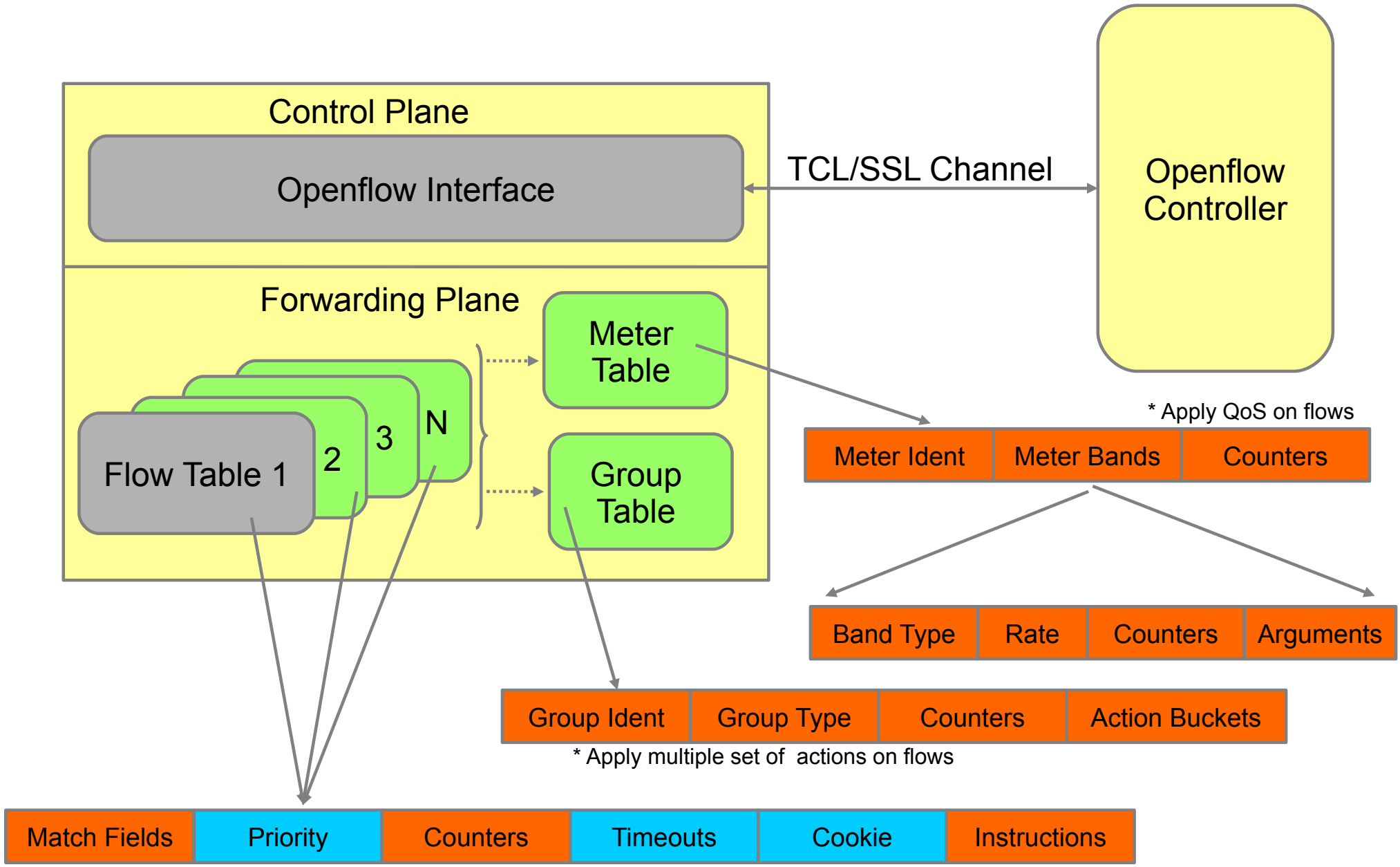
Openflow 1.3.1 Support : Main areas to analyze

- Controller- Switch communication protocol:
  - New instructions to program openflow 1.3 switch
  - New messages from the switch to controller and vice versa
  - New features/stats polling request/response
- Controller – Switch connectivity mechanism

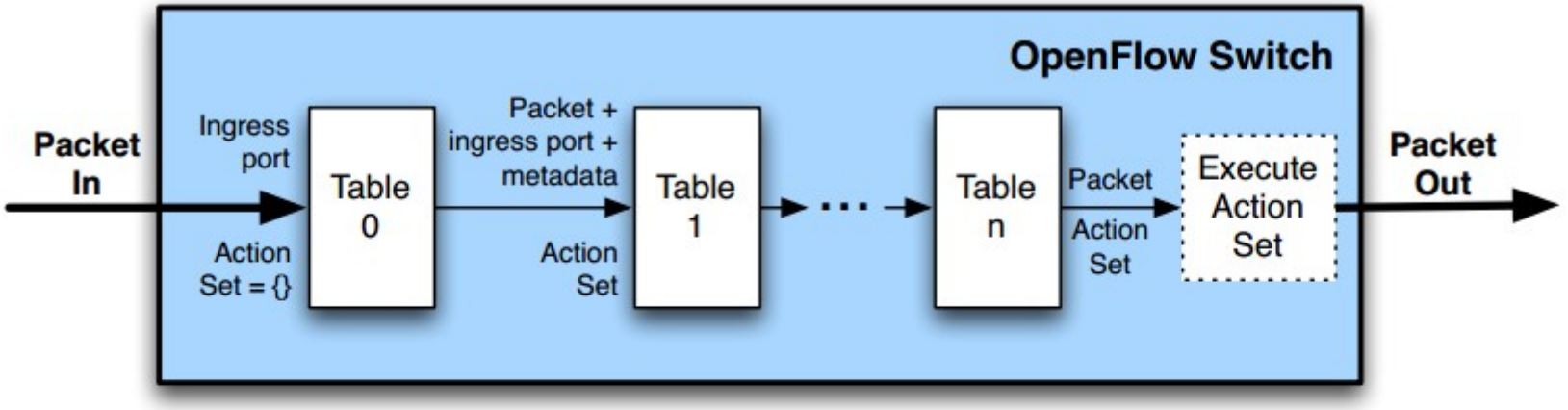
Openflow 1.0.0: Switch Hardware



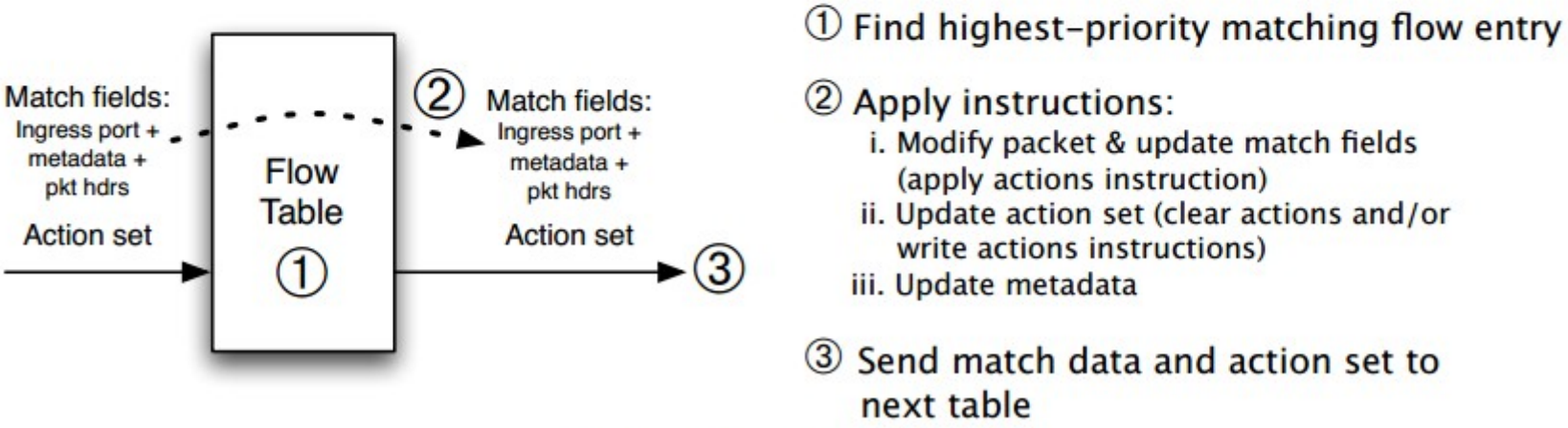
Openflow 1.3.1: Switch Hardware (Forwarding Plane)



Openflow 1.3.1 : Packet flow through the processing pipeline



(a) Packets are matched against multiple tables in the pipeline



(b) Per-table packet processing

Action Set – is associated with each packet. This set is empty by default. Instructions can modify action set. An action set contains a maximum of one action of each type.

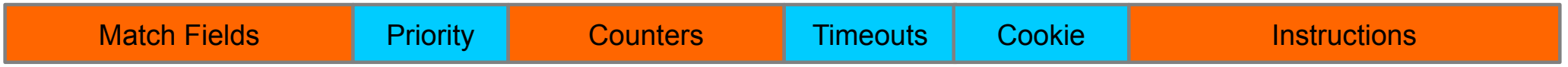
Openflow 1.0.0 : Flow Entry

- Ingress Port
- Header Fields
  - Ether Src
  - Ether dst
  - Ether type
  - VLAN Id
  - VLAN prio
  - IP Src
  - IP Dst
  - IP Proto
  - IP TOS (DSCP)
  - TP Src Port
  - TP Dst Port

- Per Table
  - Active Entries
  - Packet Lookups
  - Packet Matches
- Per Flow
  - Received Packets
  - Received Bytes
  - Duration (Seconds)
  - Duration (Nano Secs)
- Per Port
  - Received Packets
  - Transmitted Packets
  - Received Bytes
  - Transmitted Bytes
  - Receive Drops
  - Transmit Drops
  - Receive Errors
  - Transmit Errors
  - Receive Frame Errors
  - Receive Overrun Errors
  - Receive CRC Errors
  - Collisions
- Per Queue
  - Transmit Packets
  - Transmit Bytes
  - Transmit Overrun Errors

- ALL
- CONTROLLER
- LOCAL
- TABLE
- IN_PORT
- NORMAL
- FLOOD
- Enqueue
- Drop
- Modify-Field
  - VLAN Id
  - VLAN Prio
  - Pop VLAN
  - Ether Src
  - Ether Dst
  - IP Src
  - IP Dst
  - IP Tos
  - TP Src Port
  - TP Dst Port

Openflow 1.3.1 : Flow Entry



- Port
  - Ingress Port
  - Physical Input Port
- Metadata
- Header Fields
  - Ether Src
  - Ether dst
  - Ether type
  - VLAN Id
  - VLAN prio
  - IP Src
  - IP Dst
  - IP Proto
  - IP TOS (DSCP)
  - IP TOS (ECN)
  - TCP Src Port / TCP Dst Port
  - UDP Src Port / UDP Dst Port
  - SCTP Src Port / SCTP Dst Port
  - ICMP Type / ICMP Code
  - Arp Opcode
  - Arp Src IP / Arp Dst IP
  - Arp Ether Src / Arp Ether Dst
  - IPv6 Src / IPv6 Dst
  - IPv6 Flow Label
  - ICMPv6 Type / ICMPv6 Code
  - IPv6 ND Dst address
  - IPv6 ND Src Link Layer
  - IPv6 ND Dst Link Layer
  - MPLS Label
  - MPLS TC
  - MPLS BoS bit
  - PBB I-SID
  - Tunnel ID (Metadata)
  - Ipv6 Extension Header

- Per Flow Table
  - Active Entries
  - Packet Lookups
  - Packet Matches
- Per Flow
  - Received Packets
  - Received Bytes
  - Duration (Seconds)
  - Duration (Nano Secs)
- Per Port
  - Received Packets
  - Transmitted Packets
  - Received Bytes
  - Transmitted Bytes
  - Receive Drops
  - Transmit Drops
  - Receive Errors
  - Transmit Errors
  - Receive Frame Errors
  - Receive Overrun Errors
  - Receive CRC Errors
  - Collisions
  - Duration (Seconds)
  - Duration (Nano Sec)
- Per Queue
  - Transmit Packets
  - Transmit Bytes
  - Transmit Overrun Errors
  - Duration (Seconds)
  - Duration (Nano Secs)

- Meter <meter_id> – Direct packet to specified meter
- Apply-Actions <action(s)> - Apply action(s) immediately
- Clear-Actions – Clear all actions in action set
- Write-Metadata <metadata / mask > - Write masked metadata to metadata field
- Goto-Table <next-table-id> - Next table in processing

- Actions:
  - Output
    - ALL
    - CONTROLLER
    - LOCAL
    - TABLE
    - IN_PORT
    - NORMAL
    - FLOOD
  - Set-queue (Enqueue)
  - Drop
  - Group
  - Push Tag (VLAN, MPLS, PBB)
  - Pop Tag (VLAN, MPLS, PBB)
  - Set Field
    - All the Header Fields specified in Match Fields
  - Set TTL (MPLS TTL, IP TTL)
  - Decrement TTL (MPLS TTL, IP TTL)
  - Copy TTL outwards
  - Copy TTL inwards

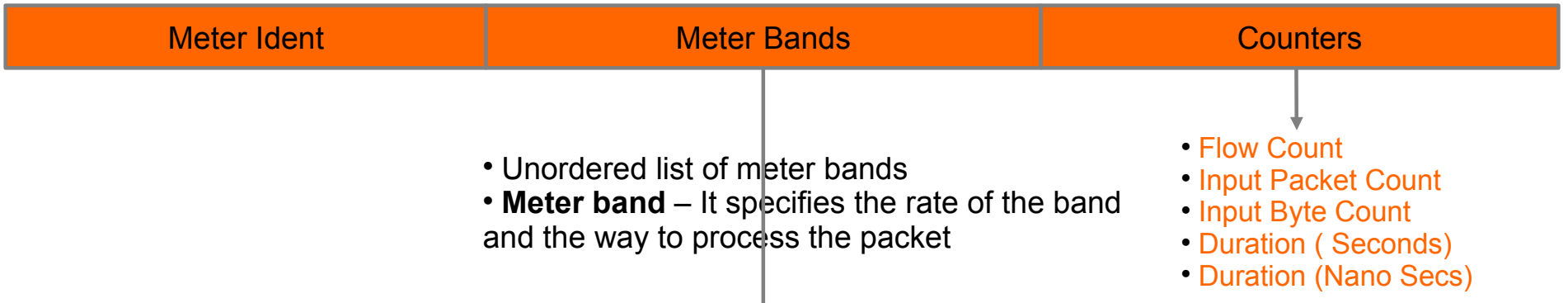
Openflow 1.3.1 : Group Table

- All – Execute all buckets in the group
- Select – Execute one bucket in the group
- Indirect – Execute the one defined bucket in this group
- Fast-failover – Execute the first live bucket

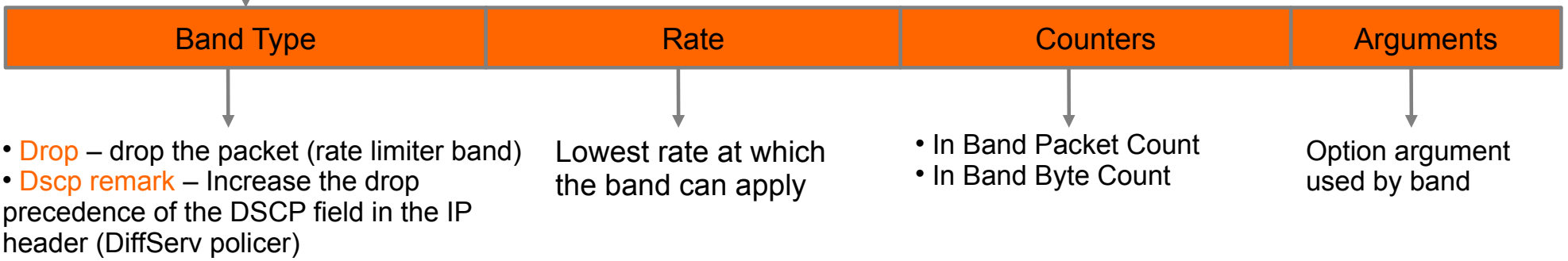
- Reference Count (Flow Entries)
- Packet Count
- Byte Count
- Duration (Seconds)
- Duration (Nano Secs)

- An ordered list of action buckets.
- **Action buckets** – Contains a set of actions to execute

Openflow 1.3.1 : Meter Table



Openflow 1.3.1 : Meter Band



Controller work items : Protocol common structures

Work Items	Opendaylight code sections to modify		
	OpenflowJ	protocol_plugin	SAL
Define new flow entry match fields (Flow Entry Slide)	√	√	√
Modify flow entry stats structure to accommodate new entries (Flow Entry Slide – Per port/queue)	√	√	X
Implement new instruction (Flow Entry Slide)	√	√	√
Implement new actions and restructure existing actions (Flow Entry Slide)	√	√	√
Implement structure of Group Table (Group Table Slide)	√	√	√
Define new group types (Group Table Slide)	√	√	√
Implement new counter/stats strcuture for group table (Group Table Slide)	√	√	X
Implement new structure for action bucket (Group Table Slide)	√	√	√
Implement new structures for Meter Table and Meter Band (Meter Table Slide)	√	√	√
Define new band types (Meter Table Slide)	√	√	√
Implement new structures for Meter Table and Meter Band counters/stats (Meter Table Slide)	√	√	X

Controller work items : Low level details (Protocol common structures)

Openflow 1.0.0 - Structure & Enums	Openflow 1.3.1 - Structure & Enums	Level of change	Backward Compatibility
Struct ofp_header	Struct ofp_header	No change	
Enum ofp_type	Enum ofp_type	Minor addition and removal	No
Struct ofp_phy_port	Struct ofp_port	Minor addition	Yes
Enum ofp_port_config	Enum ofp_port_config	Minor removal	Yes
Enum ofp_port_state	Enum ofp_port_state	Minor removal	Yes
Enum ofp_port	Enum ofp_port_no	Minor addition and removal	No
Enum ofp_port_features	Enum ofp_port_features	Minor addition	No
Struct ofp_packet_queue	Struct ofp_packet_queue	Minor addition	Yes
Enum ofp_queue_properties	Enum ofp_queue_properties	Minor addition and removal	Yes
Struct ofp_queue_prop_header	Struct ofp_queue_prop_header	No Change	Yes
Struct ofp_queue_prop_min_rate	Struct ofp_queue_prop_min_rate	No Change	Yes
	Struct ofp_prop_max_rate	New Addition	
	Struct ofp_queue_prop_experimenter	New Addition	

Controller work items : Low level details (Protocol common structures) (contd.)

Openflow 1.0.0 - Structure & Enums	Openflow 1.3.1 - Structure & Enums	Level of change	Backward Compatibility
Struct ofp_match	Struct ofp_match	Entirely different structure	No
	Enum ofp_match_type	New Addition	
	Enum ofp_oxm_class	New Addition	
	Enum ofp_vlan_id	New Addition	
	Enum ofp_ipv6exthdr_flags	New Addition	
	Struct ofp_oxm_experimenter_header	New Addition	
	Enum ofp_instruction_type	New Addition	
	Struct ofp_instruction	New Addition	
	Struct ofp_instruction_goto_table	New Addition	
	Struct ofp_instruction_write_metadata	New Addition	
	Struct ofp_instruction_actions	New Addition	
	Struct ofp_instruction_meter	New Addition	
	Struct ofp_instruction_experimenter	New Addition	
Enum ofp_flow_wildcards		Not present	Yes

Controller work items : Low level details (Protocol common structures) (contd.)

Openflow 1.0.0 - Structure & Enums	Openflow 1.3.1 - Structure & Enums	Level of change	Backward Compatibility
Enum ofp_action_type	Enum ofp_action_type	Old entries are removed and new entries are added. Enum number is not effected.	Yes
Struct ofp_action_header	Struct ofp_action_header	No Change	
Struct ofp_action_output	Struct ofp_action_output	Minor change	Yes
	Enum ofp_controller_max_len	New addition	
	Struct ofp_action_group	New addition	
Struct ofp_action_enqueue	Struct ofp_action_set_queue	Minor removal	Yes
	Struct ofp_action_mpls_ttl	New addition	
	Struct ofp_action_nw_ttl	New addition	
	Struct ofp_action_push	New addition	
	Struct ofp_action_pop_mpls	New addition	
Struct ofp_action_vlan_vid	Struct ofp_action_set_field	All the action structs are removed and new struct is introducted, which will be used to define set field actions. It will be used for all the set field actions mentioned in slide-11 (under match fields)	No
Struct ofp_action_vlan_pcp			
Struct ofp_action_dl_addr			
Struct ofp_action_nw_addr			
Struct ofp_action_nw_tos			
Struct ofp_action_tp_port			
Struct ofp_action_vendor_header		Removed	
	Struct ofp_action_experimenter_header	New addition	

Code level change requirements : Protocol common structures

- Openflowj driver:
  - Implement / modify the structures and enum as mentioned in the low level detail slides
- SAL:
  - Implement the corresponding abstraction (e.g New actions and match)
  - Implement abstractions for group and meter counters/stats data
- Protocol_plugin:
  - Implement/modify the converter code to convert SAL abstract instruction/commands to protocol specific instruction/commands (e.g. SAL (Flow) to openflowj (OFFlowMod))
  - Implement/modify stats collector code and convert to the corresponding SAL abtraction

Open Flow 1.0.0 : Controller – Switch Protocol Messages

- Features
  - Switch features
- Configuration
  - Switch configuration
  - Queue configuration
- Modify-States
  - Modify flow entry
  - Modify port
- Read-State
  - Description statistics
  - Individual flow statistics
  - Aggregate flow statistics
  - Table statistics
  - Port statistics
  - Queue statistics
  - Vendor statistics
- Send-Packet
  - Send packet out
- Barrier
  - Barrier request

- Packet_in
- Flow-Removed
- Port-status
- Error

- Hello
- Echo request
- Echo reply
- Vendor

Open Flow 1.3.1 : Controller – Switch Protocol Messages

- Features
  - Switch features
- Configuration
  - Switch configuration
  - Queue configuration
  - Flow Table configuration
- Modify-States
  - Modify flow entry
  - Modify group entry
  - Modify port
  - Modify meter
- Multi-part (Read-states)
  - Description statistics
  - Individual flow statistics
  - Aggregate flow statistics
  - Table statistics
  - Table features
  - Port statistics
  - Port description
  - Queue statistics
  - Group statistics
  - Group description
  - Group features
  - Meter statistics
  - Meter configuration statitics
  - Meter features statistics
  - Experimenter Multipart
  - Vendor statistics
- Send-Packet
  - Send packet out
- Barrier
  - Barrier request
- Role-Request
- Asynchronous-Configuration

- Packet_in
- Flow-Removed
- Port-status
- Error

- Hello
- Echo request
- Echo reply
- Experimenter

Controller work items : Controller – Switch Protocol Messages

Work Items (Refer OpenFlow 1.3.1 controller-switch protocol messages slide)	Opendaylight code sections to modify		
	openflowj	protocol_plugin	SAL
Implement flow table configuratoin message request and response structures	√	√	X
Implement flow table feature message request and response structure	√	√	√
Implement group table related message request and response structures : [description, statistics, feature, entry modification]	√	√	X
Implement meter table related message request and response structures : [statistics, configuration statistics, feature statistics, meter modification]	√	√	X
Implement experimenter multipart message request and response structures	√	√	X
Implement role-request message request and response structures	√	√	√
Implement asynchronous-configuartion message reqest and response structures	√	√	X
Modify error enum and add new error messages related to the above mentioned new request and response messages.	√	√	X
Implement symmetric experimenter request and response message structure	√	√	X
Modify existing request response messages to parse the additional details send by openflow 1.3 enabled switches. Refer low level details (controller-switch protocol messages) slides.	√	√	√

Controller work items : Low level details (Controller–Switch Protocol Messages)

Openflow 1.0.0 - Structure & Enums	Openflow 1.3.1 - Structure & Enums	Level of change	Backward Compatibility
Controller -to -Switch Messages			
Struct ofp_switch_features	Struct ofp_switch_features	Minor addition and removal	Yes
Enum ofp_capabilities	Enum ofp_capabilities	Minor addition	No
Struct ofp_switch_config	Struct ofp_switch_config	No Change	
Enum ofp_config_flags	Enum ofp_config_flags	No Change	
	Enum ofp_table	New addition	
	Struct ofp_table_mod	New addition	
	Enum ofp_table_config	New addition	
Struct ofp_flow_mod	Struct ofp_flow_mod	Minor addition	Yes
Enum ofp_flow_mod_command	Enum ofp_flow_mod_command	No Change	
Enum ofp_flow_mod_flags	Enum ofp_flow_mod_flags	Minor addition and removal	No
	Struct ofp_group_mod	New addition	
	Enum ofp_group_mod_command	New addition	
	Enum ofp_group_type	New addition	
	Enum ofp_group	New addition	
	Struct ofp_bucket	New addition	
Struct ofp_port_mod	Struct ofp_port_mod	Minor addition and removal	No
	Struct ofp_meter_mod	New addition	
	Enum ofp_meter	New addition	
	Enum ofp_meter_mod_command	New addition	
	Enum ofp_meter_flags	New addition	

Controller work items : Low level details (Controller–Switch Protocol Messages)

Controller -to -Switch Messages			
	Struct ofp_meter_band_header	New addition	
	Enum ofp_meter_band_type	New addition	
	Struct ofp_meter_band_drop	New addition	
	Struct ofp_meter_band_dscp_remark	New addition	
	Struct ofp_meter_band_experimenter	New addition	
Struct ofp_queue_get_config_request	Struct ofp_queue_get_config_request	Minor change	No
Struct ofp_queue_get_config_reply	Struct ofp_queue_get_config_reply	Minor change	No
Struct ofp_stats_request	Struct ofp_multipart_request (renamed)	Minor addition	Yes
	Enum ofp_multipart_request_flags	New addition	
Struct ofp_stats_reply	Struct ofp_multipart_reply (renamed)	Minor addition	Yes
Enum ofp_stats_types	Enum ofp_multipart_types (renamed)	Minor addition	Yes
Struct ofp_desc_stats	Struct ofp_desc	No Change	
Struct ofp_flow_stats_request	Struct ofp_flow_stats_request	Minor change and addition	No
Struct ofp_flow_stats	Struct ofp_flow_stats	Minor addition	Yes
Struct ofp_aggregate_stats_request	Struct ofp_aggregate_stats_request	Minor change and addition	No
Struct ofp_aggregate_stats_reply	Struct ofp_aggregate_stats_reply	No Change	Yes
Struct ofp_table_stats	Struct ofp_table_stats	Minor addition	Yes
	Struct ofp_table_features	New addition	
	Enum ofp_table_feature_prop_type	New addition	

Controller work items : Low level details (Controller–Switch Protocol Messages)

Controller -to -Switch Messages			
	Struct ofp_table_feature_prop_header	New addition	
	Struct ofp_table_feature_prop_instructions	New addition	
	Struct ofp_table_feature_prop_next_tables	New addition	
	Struct ofp_table_feature_prop_actions	New addition	
	Struct ofp_table_feature_prop_oxm	New addition	
	Struct ofp_table_feature_prop_experimenter	New addition	
Struct ofp_port_stats_request	Struct ofp_port_stats_request	Minor change	Yes
Struct ofp_port_stats	Struct ofp_port_stats	Minor change and addition	Yes
	Struct ofp_port	New addition	
Struct ofp_queue_stats_request	Struct ofp_queue_stats_request	Minor change	Yes
Sruct ofp_queue_stats	Struct ofp_queue_stats_request	Minor change and addition	Yes
Struct ofp_packet_out	Struct ofp_packet_out	Monor change and addition	No
	Struct ofp_group_stats_request	New addition	
	Struct ofp_group_stats	New addition	
	Struct ofp_bucket_counter	New addition	
	Struct ofp_group_desc_stats	New addition	
	Struct ofp_group_features	New addition	
	Enum ofp_group_capabilities	New addition	
	Struct ofp_meter_multipart_request	New addition	

Controller work items : Low level details (Controller–Switch Protocol Messages)

Controller -to -Switch Messages			
	Struct ofp_group_features	New addition	
	Enum ofp_group_capabilities	New addition	
	Struct ofp_meter_multipart_request	New addition	
	Struct ofp_meter_stats	New addition	
	Struct ofp_meter_band_stats	New addition	
	Struct ofp_meter_multipart_request	New addition	
	Struct ofp_meter_config	New addition	
	Struct ofp_meter_features	New addition	
	Struct ofp_experimenter_multipart_header	New addition	
	Struct ofp_role_request	New addition	
	Enum ofp_controller_role	New addition	
	Struct ofp_async_config	New addition	
Asynchronous Messages			
Struct ofp_packet_in	Struct ofp_packet_in	Minor addition	Yes
Enum ofp_packet_in_reason	Enum ofp_packet_in_reason	Minor addition	Yes
Struct ofp_flow_removed	Struct ofp_flow_removed	Minor addition	Yes
Enum ofp_flow_removed_reason	Enum ofp_flow_removed_reason	Minor addition	Yes
Struct ofp_port_status	Struct ofp_port_status	No change	
Enum ofp_port_reason	Enum ofp_port_reason	No Change	
Struct ofp_error_msg	Struct ofp_error_msg	No Change	
Enum ofp_error_type	Enum ofp_error_type	Minor addition	No

Controller work items : Low level details (Controller–Switch Protocol Messages)

Asynchronous Messages			
Enum ofp_hello_failed_code	Enum ofp_hello_failed_code	No change	
Enum ofp_bad_request_code	Enum ofp_bad_request_code	Minor addition and change	No
Enum ofp_bad_action_code	Enum ofp_bad_action_code	Minor addition and change	Yes
	Enum ofp_bad_instruction_code	New addition	
	Enum ofp_bad_match_code	New addition	
Enum ofp_flow_mod_failed_code	Enum ofp_flow_mod_failed_code	Minor addition and removal	No
	Enum ofp_group_mod_failed_code	New addition	
Enum ofp_port_mod_failed_code	Enum ofp_port_mod_failed_code	Minor addition	Yes
	Enum ofp_table_mod_failed_code	New addition	
Enum ofp_queue_op_failed_code	Enum ofp_queue_op_failed_code	No change	
	Enum ofp_switch_config_failed_code	New addition	
	Enum ofp_role_request_failed_code	New addition	
	Enum ofp_meter_mode_failed_code	New addition	
	Enum ofp_table_feature_failed_code	New addition	
	Struct ofp_error_experimenter_msg	New addition	
Symmetric Message			
	Struct ofp_hello	New addition	
	Enum ofp_hello_elem_type	New addition	
	Struct ofp_hello_elem_header	New addition	
	Struct ofp_hello_elem_versionbitmap	New addition	
	Struct ofp_experimenter_header	New addition	

Code level change requirements : Controller – Switch Protocol Messages

- Openflowj driver:
  - Implement / modify the structures and enum as mentioned in the low level detail slides
- SAL:
  - Most of the required abstraction will get implemented while adding common structures to the openflowj and implemented their relevant abstractions in SAL. Few additional abstractions like Controller Role request need to be implemented. This abstraction can be used by controller HA/clustering services.
- Protocol_plugin:
  - Implement the code to execute newly defined instruction/command to the switch.
  - Implement stats collector code to request stats for the newly defined elements.

Controller – Switch connection mechanism comparison:

Open flow 1.0.0	Open flow 1.3.1
Single connection between switch and controller pair (Main connection)	Openflow 1.3.1 also specify the same.
	Switch can trigger auxilary connection to the same controller.
	Switch can send different type of packets through different auxilary connections or main connection
	Switch should get response of the openflow message from the same auxilary connection from which it was sent
	Auxilary connection are only allowed once main connection is setup
	Auxilary connection can use any protocol for connection (UDP, DTLS)
	Each auxilary connection setup takes the same steps as main connection
	Auxilary connection should have same destination controller IP and port unless specified otherwise by switch
The switch and controller mutually authenticate by exchanging certificates signed by site-specific private-key.	Similar mechanism is specified for 1.3.1 specs. Although this is for main conenction and not auxilary connections.
Connection setup process usages openflow version number for negotiating the protocol version to be used.	Similar mechanism is specified for 1.3.1 specs. Although this spec introduced new version negotiation mechanism based on the OFPHET_VERSIONBITMAP hello element. If that fails it falls back to the version number based negotiation.
	Controller can change its role for specific switch from master contoller to slave controller or viceversa by sending role change request to the switch.

Controller work items : Controller – Switch connection mechanism

Work Items	Opendaylight code sections to modify		
	openflowj	protocol_plugin	SAL
Implement addition process of version negotiation based on OFPHET_VERSIONBITMAP at the time of handshake.	X	√	X
Implement auxilary connection mechanism on controller side.	X	√	X
Implement role change request/response open flow messages. This will be required when we design controller HA solution and that requires specific controller in specific role at any point of time.	X	√	√

Code level change requirements : Controller – Switch connection mechanism

- Controller-switch connection mechanism is handled by protocol_plugin code, so this is the only place that requires major code changes.

- Complex parts to handle in this mechanism is
  - Sending response on the same channel from where request came in
  - Implementing the synchronous messages across different auxilary connections using barrier messages
  - Using different mechanism in the presence of openflow 1.0.0 and openflow 1.3.1 enabled switches

Backup slides

(Conceptual Changes)

Conceptual changes - Openflow 1.3.1 :

(Something to ponder before designing openflow 1.3.1 based applications)

Conceptual changes in flow rule and action
Support for multiple flow tables is introduced (Slide -12). Matching always start from the first table (table-id=0) and then it can continue to next tables.
New instruction set is introduced. *Apply-Action* is the instruction which is similar to specifying actions for the flow entry in openflow 1.0. For example OF 1.0 : Match-timeouts:Action1, Action2, Action3 OF 1.3 : Match-timeouts: *Action-Apply* <Action1, Action2, Action3> This action execute immediately and modify the packet. So if you have Goto Instruction after this instructoin in flow entry instruction set, next table will get modified packet. Rest all instructions won't modify packet before forwarding it to next table. This instruction execute the action in the order they are specified.
In multiple table notion, every packet_in has associated action set. Each instruction (except *Apply-Action* and *Goto*) will add/remove actions to the action set accordingly. This action set gets executed in two scenarnio a) When you reach the last table. Last table won't allow flow entries with Goto Instruction, because processing through table is only allowed in forward direction (E.g in table X you can't specify instruction 'Goto X-2', next table should always be >X). b) When flow entry don't have Goto instruction in its flow entry instruction, it means that table will act as a last table for that specific flow entry.
An action set containsl a maximum of one action of each type. When multiple actions of the same type are required, the *Apply-Actions* instruction should be used.
Support for passing metadata information between the tables.
Concept of logical port introduced (e.g. Link aggregation groups, tunnels, loopback interfaces). Every packet_in reaches to the controller will have logical_port (ingress port) and physical port number in the packet_in message.
Instruction set of each flow entry can have at most 1 instruction of each type. So if you want to execute same instruction more then once, you have to forward the packet to next table and execute it there.

Conceptual changes - Openflow 1.3.1 (contd):

Conceptual changes in flow rule and action
Execution order of instructions in instruction set is defined in spec. Instructions won't get executed in the order they were added into the flow entry instruction set. Order of execution is : Meter → Apply-Actions → Clear-Actions → Write-Metadata → Goto
Bahavior in case of no matching flow rule entry found in table for packet-in: OF 1.0 : Send it to controller OF 1.3 : Drop the packet. To send this packet to controller instead of dropping, every table should have one flow entry (named *table miss flow entry*) that has all match field wildcarded and priority =0. Instruction (action = Controller) should be set for the flow rule entry to send the packet to the controller. This rule entry is same as normal flow entry controller installs to the switch. Controller can install it and remove it and it can expire as well (its not a default entry in table).
Support for group table is introduced. Group table enables execution of multiple actions from single group for multiple flow entries from different tables. Each group table entry can have *ordered* list of zero or more action bucket associated with it. As of now only two required (all , indirect) and two optional (select, fast failover) group types are specified. • All – Execute all the buckets associated with the group entry. This group is used for multicast or broadcast forwarding. The bucket is effectively cloned for each bucket, one packet is processed for each bucket of the group. • Indirect – Execute the one defined bucket in the group. This group supports only a single bucket. Allow multiple flow entries or groups to point to a commond group identifier supporting faster, more efficient convergence (e.g next hops for IP forwarding)
Actions in action bucket get executed in the following order (irrespective of their oder in the bucket) : Copy TTL inward → pop → psuh-MPLS → push-PBB → push-VLAN → copy TTL outward → decrement TTL → set → qos → group or output
If group action and output action both are specified, group action will take precedence and output action will be ignored. In case of absense of both of this actions packet will be dropped.

Conceptual changes - Openflow 1.3.1 (contd):

Conceptual changes in flow rule and action
Any group entry bucket can have group action referring to another group entry in the table. The execution of groups is recursive if the switch supports it.
Newly pushed tags should always be inserted as the outermost tag in the outermost valid location for that tag. E.g a new VLAN tag is pushed, it should be the outermost tag inserted, immediately after the Ethernet header and before other tags.
When multiple push actions are added to the action set of the packet, they apply to the packet in the order defined by the action set rules [MPLS → PBB → VLAN]
A switch may support checking that no loop is created while chaining groups: if a group mod is sent such that a forwarding loop would be created, the switch must reject the gourp mod and must send an error messages.
New controller-switch messages
New Role-Request messages are introduced. These are used by the controller to set the role of its OpenFlow channel, or query that role. This is mostly usefule when the switch connects to multiple controllers.
New Asynchronous-Configuration messages are introduced. These messages are used by the controller to set an additional filter on the asynchronous messages that it wants to receive on its open flow channel or to query that filter. This is mostly useful when the switch connects to multiple controllers.
Experimenter message are introduced and motivation behind this was to provide staging are for features meant for future open flow revisions.
Message execution ordering at switch level can be ensured through the use of barrier messages. In the absence of barrier messages, switches may arbitrarily reorder messages to maximize performance; hence,controllers should not be depend on a specific processing order. Although its present in 1.0 as well, but not sure how much are we leveraging it.

Conceptual changes - Openflow 1.3.1 (contd):

New controller-switch messages
On flow rule *Add* request to switch, if flow entry with identical match fields and proirity already resides in the request table, then that entry, including its duration, must be cleared from the table, and the new flow entry added. If the OFPFF_RESET_COUNTS flag is set, the flow entry counters must be cleared, otherwise they should be copied from the replaced flow entry.
No flow-removed message is generated for the flow entry eliminated as part of an add request.
For flow entry M*odify* requests, if a matching entry exists in the table, the instrcutions field of this entry is updated with the value from the request, whereas its cookie, idle_timeout, hard_timeout, flags, counters and duration fields are left unchanged. If the OFPFF_RESET_COUNTS flag is set, the flow entry counters must be cleared.
Delete commands can use the OFPTT_ALL value for table-id to indicate that matching flow entries are to be deleted from all flow tables.
Controller – Switch connection mechanism
New auxilary connection mechanism is introduced, where there can be additional connection between same switch and same controller apart from main connection. All the connection should use the same IP and port unless specified othewise by switch. Although this won't effect designing of application as such, but it would be good if we can leverage it in some way.