A Crash Course in OpenFlow 1.1

Rob Sherwood
August 2011
rob.sherwood@bigswitch.com

big switch networks

Talk Summary

- Background and Assumptions
  - “OpenFlow 1.1 is for WANs”
- Delta between 1.0 and 1.1
  - New features, clarifications, spec changes
- Adoption (or lack thereof)
- Known issues
  - Next steps towards OpenFlow 1.2+

Background

- Assumes: familiar with OpenFlow 1.0
- OpenFlow 1.0 was developed for campus networks, e.g., GENI with slicing
- OpenFlow 1.1 was targeted at WANs
    - Took over a year to specify
    - Driven by a small but influential group
- Backwards compatibility was NOT a goal

Target Use Cases

- Better flow table usage
    - n routes * m policies == too many flow_mods
- Fast failover (faster than controller latency)
- Multi-path forwarding, e.g., ECMP
- Support for new match types
- Litany of smaller features/concerns
- Large audience requires better overall spec clarity

OpenFlow 1.0 to 1.1

- This talk divides the differences into:
  - Complex new features
  - Simple new features
  - Various sundry changes
  - Spec clarifications
  - What was not added (... and why)

Complex New Features: Summary

- Multiple tables
  - Instructions vs. actions sets
- Group table
  - Action buckets
- Match is now an extensible TLV (sort of)

Multiple Tables: Goals

- ASICs have multi-stage processing pipelines
  - OF1.0 abstracts this all away to one table
- As a result, most firmware implementations only a small subset of hardware, e.g., TCAM
- Goal: better expose underlying hardware
  - Give programmer more precise control
  - Solve: Cartesian product of flow entries

Multi-Table: Challenges

- Need a simple model to describe all ASICs
- Diverse capabilities
    - # pipeline stages
    - state between stage, legal transitions
    - support resubmit? (for tunnel decap)
- Feature negotiation is pathological
    - intra-ASIC loops; depends on actions

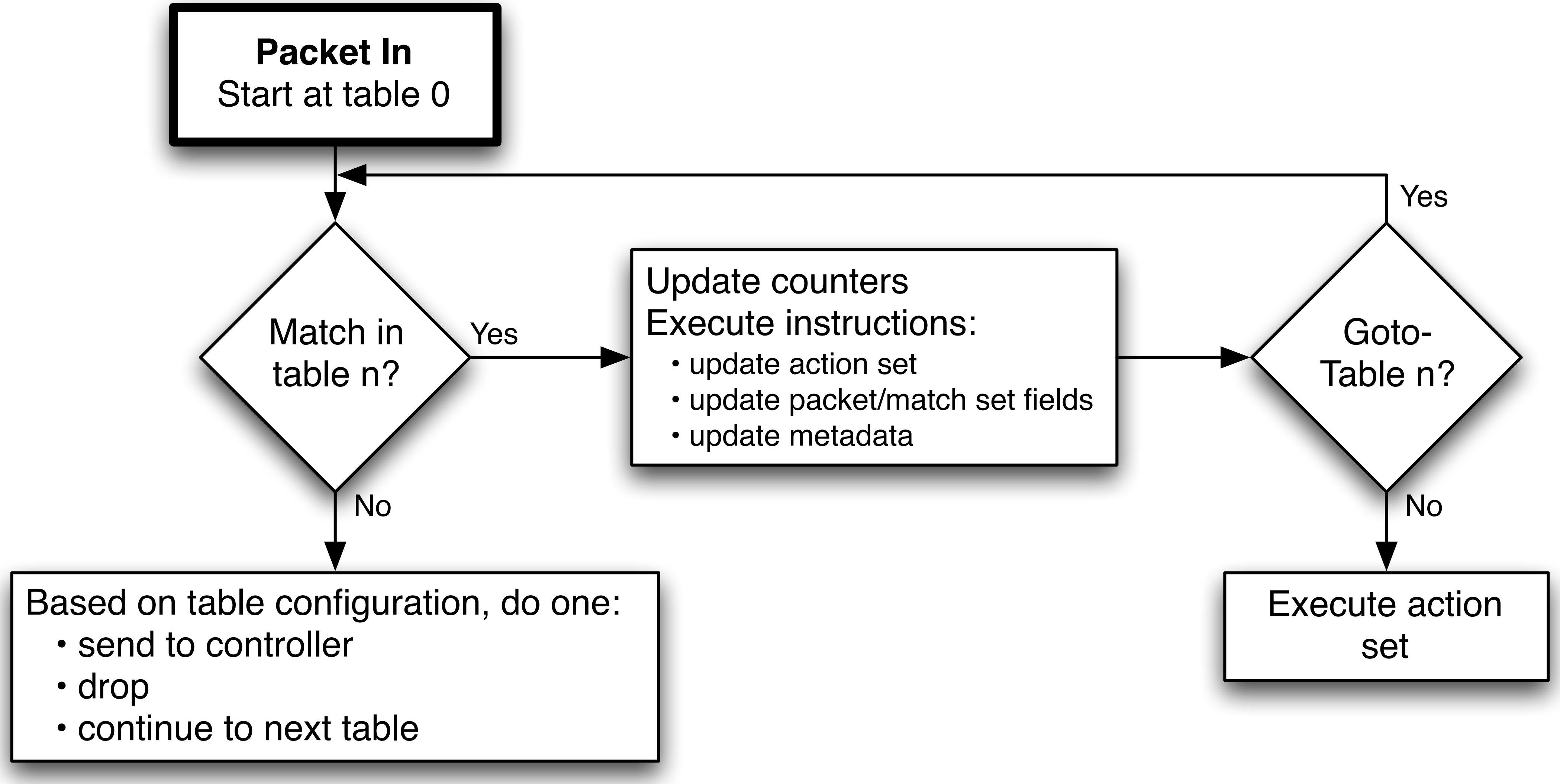
Multi-Table: Solution

- Switch exposes n tables
  - n could equal one!
- Incomplete online negotiation: too hard!
  - Assumes controller writer has OOB info
  - Switch can always say “unsupported”
- Per-table “miss” and match capabilities
- Introduce instructions and action set

Multi-Table: Instructions

- Instructions: goto-table n, record metadata, change action set, apply current actions set to packet
- Instructions affect processing pipeline, state
  - actions only affect the packet (as in 1.0)
- Actions are now a set, not a list
  - only one action of each type is allowed per packet -- closer to ASIC capabilities
  - use group table (next) to send multi-port

Multi-Table: Packet Flow



Packet In
Start at table 0

Match in table n?

Yes

Update counters
Execute instructions:
- update action set
- update packet/match set fields
- update metadata

Goto-Table n?

Yes

No

Based on table configuration, do one:
- send to controller
- drop
- continue to next table

No

Execute action set

Figure 3 from OF1.1 spec

Group Tables

- Short story: actions indirection layer
  - Added a "send to group XXX" action
- Each group is a list of action buckets
- Action bucket: a list of actions or groups
- Can create chains of action buckets
  - e.g., ECMP across links with fast failover
  - ...or even action bucket loops (!!)

Group Table: Example Uses

- Type all: execute all buckets in list
  - e.g., multi-cast groups, Spanning Tree port lists
- Type select: execute a single bucket, chosen by "switch computed selection algorithm"
  - e.g., a hash on packet 5-tuple for ECMP
- Type fast-failover: execute first live bucket
  - as managed by the switch via, e.g., BFD
- Selection algorithm, liveness criteria configured out-of-band

ofp_match is now a TLV

- Allows adopters to define new match fields
  - e.g., IPv6, FiberChannel, etc.
- Type=0 is a OF1.0-like fixed-length block
  - added support for MPLS, metadata, etc.
- No other types defined :-(
- But: can’t mix official+non-standard types
  - and assert()’s in openflow.h are wrong
  - Likely addressed in OF1.2: e.g., NXM proposal

Simple New Features

- Maskable ethernet src/dst addresses
  - e.g., for PortLand-like addressing schemes
- MPLS support: match + push/pop/swap/ttl
- VLAN QinQ support
  - Can only match outer tag
- IP TTL decrement + ECN actions added
- Maskable cookies

Litany of Other Changes

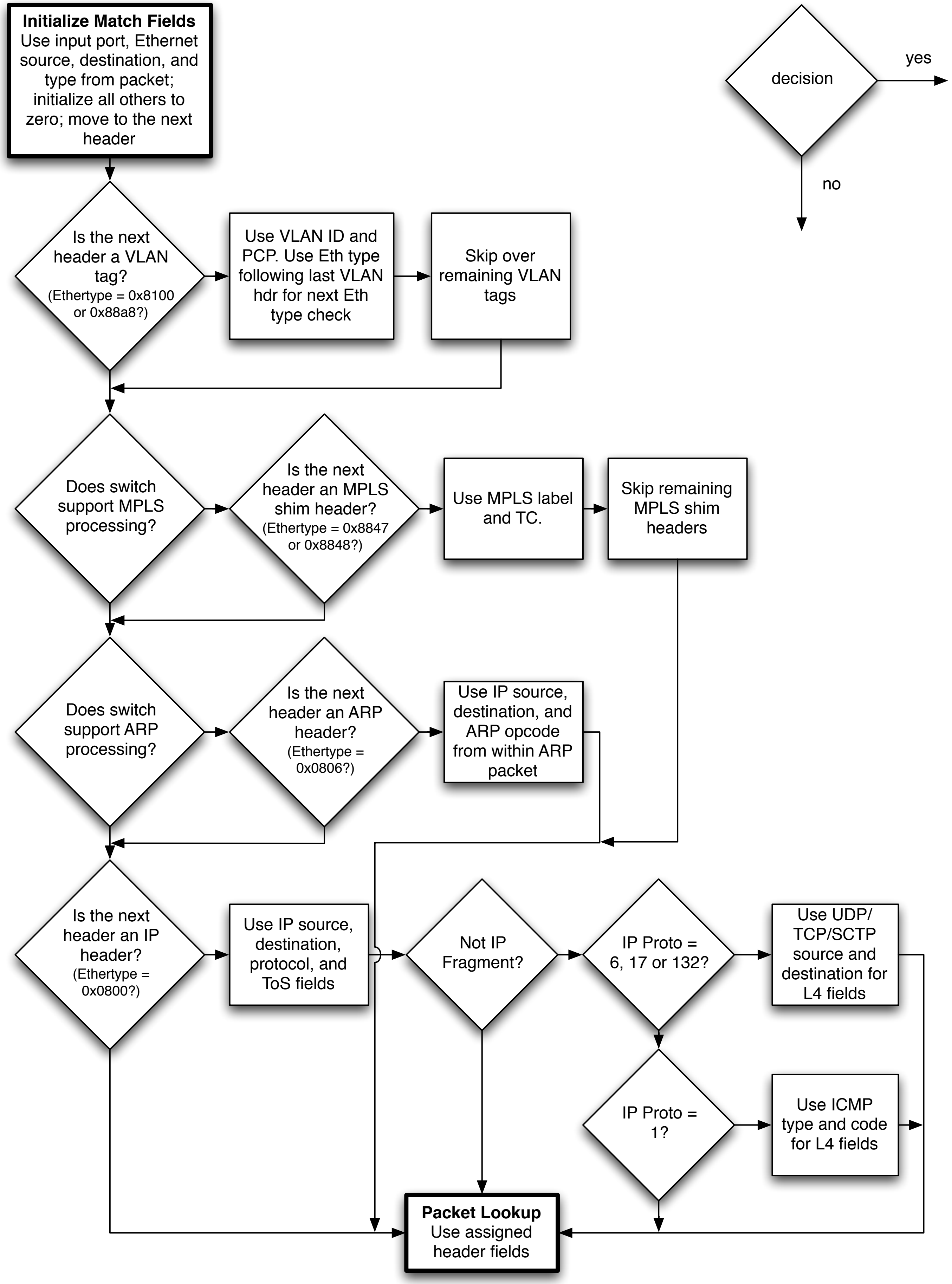
- Port IDs are now 32-bit fields
- NO_FLOOD bit can’t be controlled (!!)
- VLAN actions rewritten: push/pop/swap
- s/VENDOR/EXPERIMENTER/g
- Lots of constants renamed, reordered
- Many messages re-factored
  - e.g., flow_mod takes a list of instructions

Spec Clarifications

- Explicit packet processing model (next)
- (Partial) definition of hybrid switch
- OFPC_MODIFY vs. OFPC_ADD
  - modify is no longer an implicit add
- SSL/TLS control channel optional
  - better match to de facto use

Figure 4: How to map a packet to an ofp_match:

Main point: lots of overloaded fields to work around inflexible match.



Initialize Match Fields
Use input port, Ethernet source, destination, and type from packet; initialize all others to zero; move to the next header

Is the next header a VLAN tag?
(Ethertype = 0x8100 or 0x88a8?)

Use VLAN ID and PCP. Use Eth type following last VLAN hdr for next Eth type check

Skip over remaining VLAN tags

Does switch support MPLS processing?

Is the next header an MPLS shim header?
(Ethertype = 0x8847 or 0x8848?)

Use MPLS label and TC.

Skip remaining MPLS shim headers

Does switch support ARP processing?

Is the next header an ARP header?
(Ethertype = 0x0806?)

Use IP source, destination, and ARP opcode from within ARP packet

Is the next header an IP header?
(Ethertype = 0x0800?)

Use IP source, destination, protocol, and ToS fields

Not IP Fragment?

IP Proto = 6, 17 or 132?

Use UDP/ TCP/SCTP source and destination for L4 fields

IP Proto = 1?

Use ICMP type and code for L4 fields

Packet Lookup
Use assigned header fields

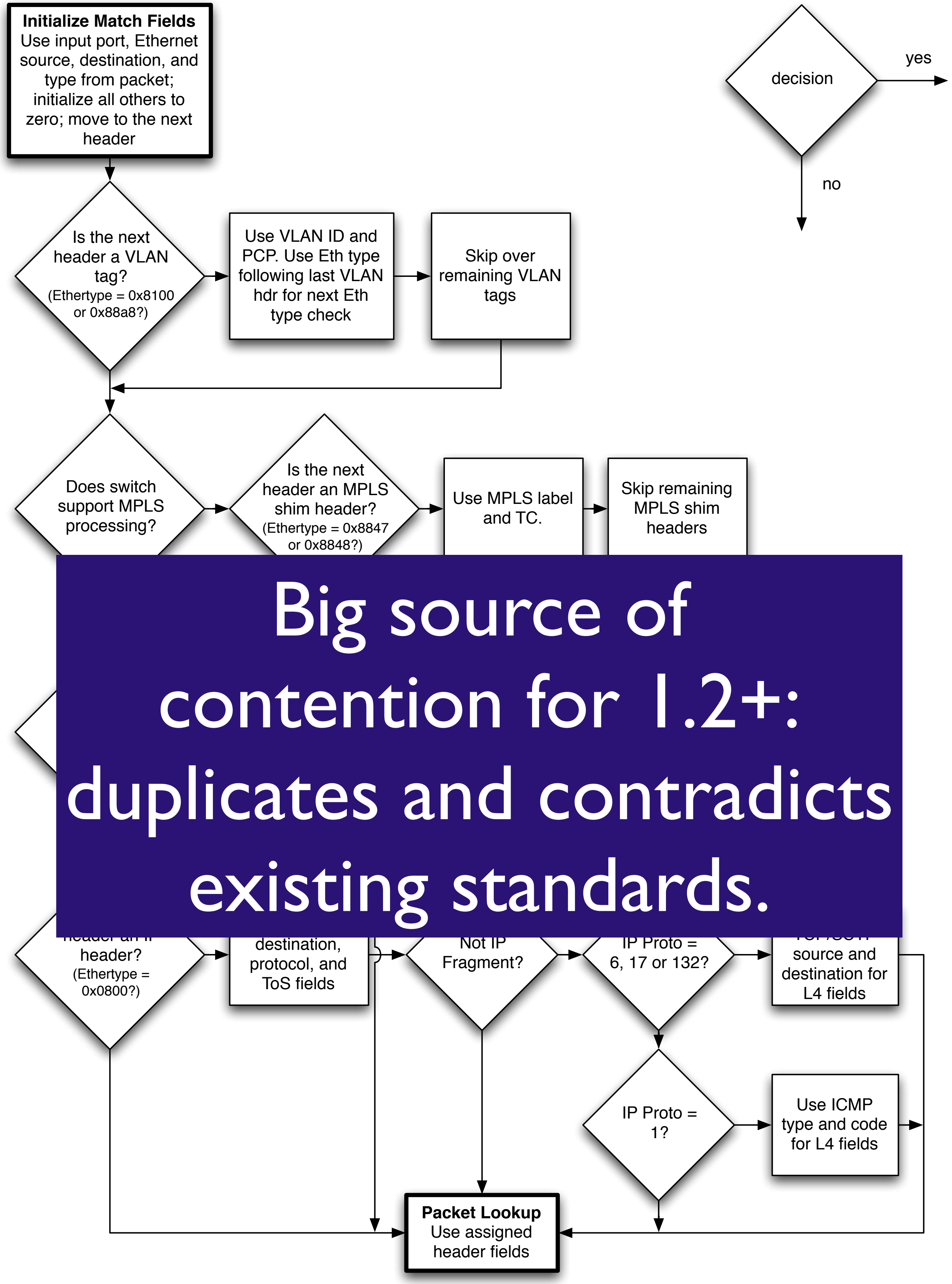
decision

yes

no

Figure 4: How to map a packet to an ofp_match:

Main point: lots of overloaded fields to work around inflexible match.


Initialize Match Fields
Use input port, Ethernet source, destination, and type from packet; initialize all others to zero; move to the next header
decision
yes
no
Is the next header a VLAN tag? (Ethertype = 0x8100 or 0x88a8?)
Use VLAN ID and PCP. Use Eth type following last VLAN hdr for next Eth type check
Skip over remaining VLAN tags
Does switch support MPLS processing?
Is the next header an MPLS shim header? (Ethertype = 0x8847 or 0x8848?)
Use MPLS label and TC.
Skip remaining MPLS shim headers
header an IP header? (Ethertype = 0x0800?)
destination, protocol, and ToS fields
Not IP Fragment?
IP Proto = 6, 17 or 132?
source and destination for L4 fields
IP Proto = 1?
Use ICMP type and code for L4 fields
Packet Lookup
Use assigned header fields

Big source of contention for 1.2+: duplicates and contradicts existing standards.

Not Added to OF1.1

- Tunneling: use virtual ports instead
  - configure out-of-band
- Configuration protocol
  - active debate in ONF working group
- Per-flow rate limiter action
  - personal pet peeve - hardware support exists!
  - really useful for OFPP_CONTROLLER

Adoption

- OF reference switch did not implement 1.1
  - code too complex to be a reference, too slow to be deployable
  - Ericsson just released OF1.1 reference (yay!)
- No OVS support (not even planned?)
- OFPS: implemented all features but group table
  - Python-based switch by Dan Talayco and myself
- EZChip NPU has an 1.1 implementation
  - AFAIK, only public “hardware”-based 1.1 switch

Known Issues (1/2)

- Full multi-tables unimplementable on existing hardware
  - Most tables have limited capabilities
    - e.g., L2-only table
  - Big increase to controller complexity
  - ...don’t even get me started on FlowVisor
- “Extensible” part of match unspecified
  - still no IPv6! planned fix in OF1.2

Known Issues (2/2)

- No controller support for 1.1
  - openflow.jar would need a rewrite
  - “hacked” nox support from Ericsson
- Still very ethernet-centric
  - No way to describe MPLS or IP-only box
- Too many things punted to OOB configuration protocol

Conclusions

- OpenFlow 1.1 solves real issues from 1.0
  - Efficient table use, ECMP, fast-failover
  - MPLS-support, VLAN QinQ
- Not (yet?) adopted for a variety of reasons
  - reasons still being debated...
- OF1.2 will hopefully address some issues