Data Structure

Michael

2017

Data Structure

VPP16.09

```
typedef struct _vlib_node_registration
{
  vlib_node_function_t *function;
  char *name;
  char *sibling_of;
  u32 index;
  vlib_node_type_t type;
  char **error_strings;
  format_function_t *format_buffer;
  unformat_function_t *unformat_buffer;
  format_function_t *format_trace;
  unformat_function_t *unformat_trace;
  u8 *(*validate_frame) (struct vlib_main_t * vm,
                                                  struct vlib_node_runtime_t *,
                                                  struct vlib_frame_t * f);
  void *runtime_data;
  u16 process_log2_n_stack_bytes;
  u8 runtime_data_bytes;
  u8 state;
  u16 flags;
  u16 scalar_size, vector_size;
  u16 n_errors;
  u16 n_next_nodes;
  struct _vlib_node_registration *next_registration;
  char *next_nodes[];
} vlib_node_registration_t;
```

```
typedef struct vlib_node_t
{
  vlib_node_function_t *function;
  u8 *name;
  u32 name_elog_string;
  vlib_node_stats_t stats_total;
  vlib_node_stats_t stats_last_clear;
  vlib_node_type_t type;
  u32 index;
  u32 runtime_index;
  void *runtime_data;
  u16 flags;
  u8 state;
  u8 runtime_data_bytes;
  u16 n_errors;
  u16 scalar_size, vector_size;
  u32 error_heap_handle;
  u32 error_heap_index;
  char **error_strings;
  char **next_node_names;
  u32 *next_nodes;
  char *sibling_of;
  uword *sibling_bitmap;
  u64 *n_vectors_by_next_node;
  uword *next_slot_by_node;
  uword *prev_node_bitmap;
  u32 owner_node_index, owner_next_index;
  format_function_t *format_buffer;
  unformat_function_t *unformat_buffer;
  format_function_t *format_trace;

  u8 *(*validate_frame) (struct vlib_main_t * vm,
                                                  struct vlib_node_runtime_t *,
                                                  struct vlib_frame_t * f);
  u8 *state_string;
} vlib_node_t;
```

Copyright© 2017. All rights reserved.

Data Structure(Cont')

VPP16.09

```
typedef struct
{
  vlib_node_t **nodes;
  uword *node_by_name;
  u32 flags;
  vlib_node_runtime_t *nodes_by_type[VLIB_N_NODE_TYPE];
  u32 *pending_interrupt_node_runtime_indices;
  u32 polling_threshold_vector_length;
  u32 interrupt_threshold_vector_length;
  vlib_next_frame_t *next_frames;
  vlib_pending_frame_t *pending_frames;
  timing_wheel_t timing_wheel;
  vlib_signal_timed_event_data_t *signal_timed_event_data_pool;
  u32 *data_from_advancing_timing_wheel;
  u64 cpu_time_next_process_ready;
  vlib_process_t **processes;
  u32 current_process_index;
  vlib_pending_frame_t *suspended_process_frames;
  void **recycled_event_data_vectors;
  u32 input_node_counts_by_state[VLIB_N_NODE_STATE];
  uword *frame_size_hash;
  vlib_frame_size_t *frame_sizes;
  f64 time_last_runtime_stats_clear;
  vlib_node_registration_t *node_registrations;
} vlib_node_main_t;
```

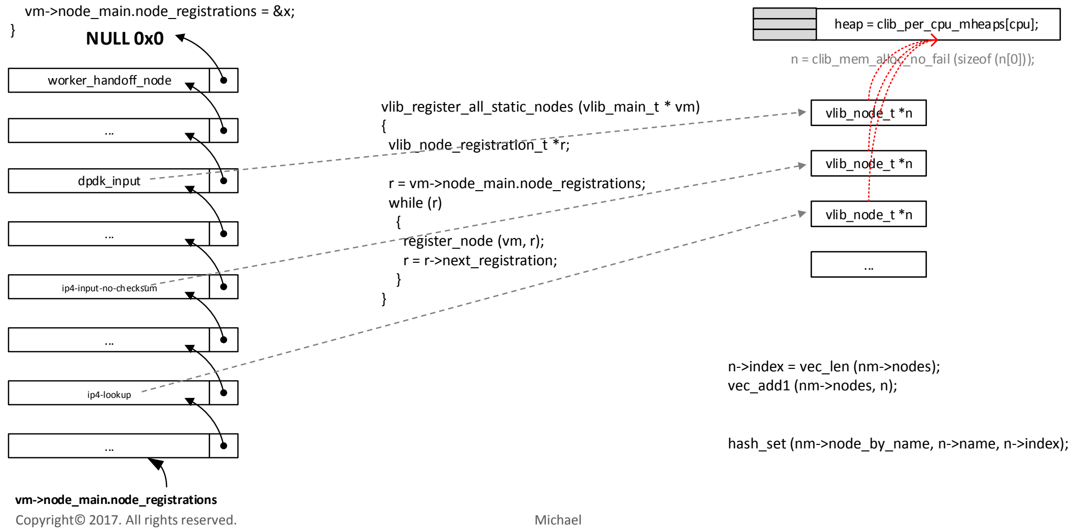
```
typedef struct vlib_node_runtime_t
{
  vlib_node_function_t *function;
  vlib_error_t *errors;
  u32 clocks_since_last_overflow;
  u32 max_clock;
  u32 max_clock_n;
  u32 calls_since_last_overflow;
  u32 vectors_since_last_overflow;
  u32 next_frame_index;
  u32 node_index;
  u32 input_main_loops_per_call;
  u32 main_loop_count_last_dispatch;
  u32 main_loop_vector_stats[2];
  u16 flags;
  u16 state;
  u16 n_next_nodes;
  u16 cached_next_index;
  u16 cpu_index;
  uword runtime_data[(128
                            - 1 * sizeof (vlib_node_function_t *)
                            - 1 * sizeof (vlib_error_t *)
                            - 11 * sizeof (u32)
                            - 5 * sizeof (u16)) / sizeof (uword)];
} vlib_node_runtime_t;
```

Copyright© 2017. All rights reserved.

Data Update

VPP16.09

```
VLIB_REGISTER_NODE
{
  vlib_main_t * vm = vlib_get_main();
  x.next_registration = vm->node_main.node_registrations;
  vm->node_main.node_registrations = &x;
}
```



```
vlib_register_all_static_nodes (vlib_main_t * vm)
{
  vlib_node_registration_t *r;

  r = vm->node_main.node_registrations;
  while (r)
   {
     register_node (vm, r);
     r = r->next_registration;
   }
}
```

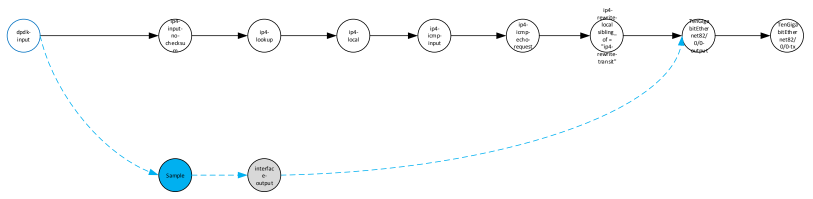
```
n->index = vec_len (nm->nodes);
vec_add1 (nm->nodes, n);

hash_set (nm->node_by_name, n->name, n->index);
```

Copyright© 2017. All rights reserved.

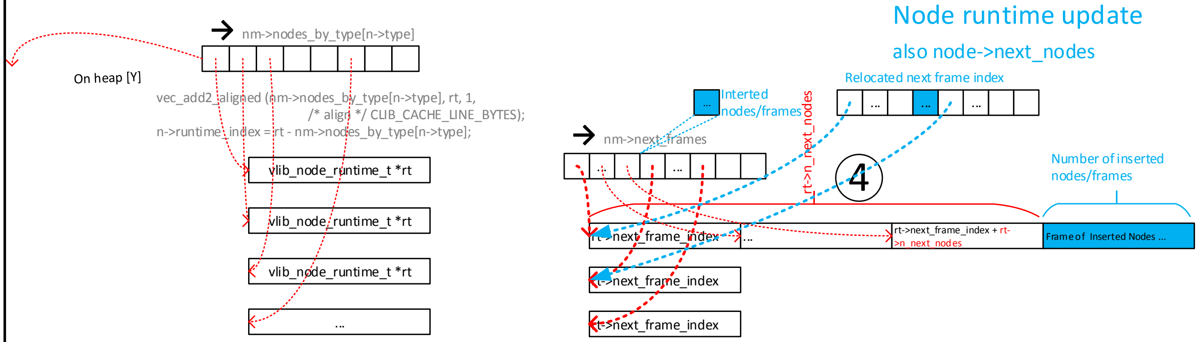
Data Update(Cont')

VPP16.09

- Runtime node
- Next_frame
- Add sample node on runtime



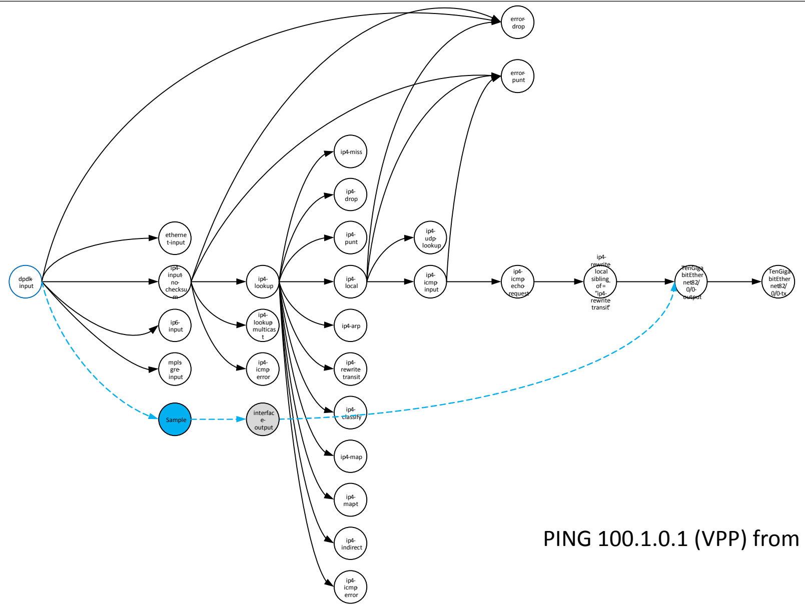
dpdk-input
ip4-input-no-checksum
ip4-lookup
ip4-local
ip4-icmp-input
ip4-icmp-echo-request
ip4-rewrite-local sibling_of = "ip4-rewrite-transit"
TenGigabitEthernet82/0/0-output
TenGigabitEthernet82/0/0-tx
Sample
interface-output



```
vec_add2_aligned (nm->nodes_by_type[n->type], rt, 1,
                  /* align */ CLIB_CACHE_LINE_BYTES);
n->runtime_index = rt - nm->nodes_by_type[n->type];
```

Register Node and Runtime Graph

VPP16.09



PING 100.1.0.1 (VPP) from 100.1.0.2(Host)

Register Node and Runtime Graph(Cont’)

```
/* *INDENT-OFF* */
VLIB_REGISTER_NODE (dpdk_input_node) = {
  .function = dpdk_input,
  .type = VLIB_NODE_TYPE_INPUT,
  .name = "dpdk-input",

  /* Will be enabled if/when hardware is detected. */
  .state = VLIB_NODE_STATE_DISABLED,

  .format_buffer = format_ethernet_header_with_length,
  .format_trace = format_dpdk_rx_dma_trace,

  .n_errors = DPDK_N_ERROR,
  .error_strings = dpdk_error_strings,

  .n_next_nodes = DPDK_RX_N_NEXT,
  .next_nodes = {
    [DPDK_RX_NEXT_DROP] = "error-drop",
    [DPDK_RX_NEXT_ETHERNET_INPUT] = "ethernet-input",
    [DPDK_RX_NEXT_IP4_INPUT] = "ip4-input-no-checksum",
    [DPDK_RX_NEXT_IP6_INPUT] = "ip6-input",
    [DPDK_RX_NEXT_MPLS_INPUT] = "mpls-gre-input",
  },
};
```

```
VLIB_REGISTER_NODE
(ip4_input_no_checksum_node,static) = {
  .function = ip4_input_no_checksum,
  .name = "ip4-input-no-checksum",
  .vector_size = sizeof (u32),

  .n_next_nodes = IP4_INPUT_N_NEXT,
  .next_nodes = {
    [IP4_INPUT_NEXT_DROP] = "error-drop",
    [IP4_INPUT_NEXT_PUNT] = "error-punt",
    [IP4_INPUT_NEXT_LOOKUP] = "ip4-lookup",
    [IP4_INPUT_NEXT_LOOKUP_MULTICAST] = "ip4-lookup-
multicast",
    [IP4_INPUT_NEXT_ICMP_ERROR] = "ip4-icmp-error",
  },

  .format_buffer = format_ip4_header,
  .format_trace = format_ip4_input_trace,
};
```

```
VLIB_REGISTER_NODE (ip4_lookup_node) = {
  .function = ip4_lookup,
  .name = "ip4-lookup",
  .vector_size = sizeof (u32),

  .format_trace = format_ip4_lookup_trace,

  .n_next_nodes = IP4_LOOKUP_N_NEXT,
  .next_nodes = IP4_LOOKUP_NEXT_NODES,
};
```

Register Node and Runtime Graph(Cont’)

```
VLIB_REGISTER_NODE (ip4_local_node,static) = {
  .function = ip4_local,
  .name = "ip4-local",
  .vector_size = sizeof (u32),

  .format_trace = format_ip4_forward_next_trace,

  .n_next_nodes = IP_LOCAL_N_NEXT,
  .next_nodes = {
    [IP_LOCAL_NEXT_DROP] = "error-drop",
    [IP_LOCAL_NEXT_PUNT] = "error-punt",
    [IP_LOCAL_NEXT_UDP_LOOKUP] = "ip4-udp-lookup",
    [IP_LOCAL_NEXT_ICMP] = "ip4-icmp-input",
  },
};
VLIB_REGISTER_NODE (ip4_icmp_input_node,static) = {
  .function = ip4_icmp_input,
  .name = "ip4-icmp-input",

  .vector_size = sizeof (u32),

  .format_trace = format_icmp_input_trace,

  .n_errors = ARRAY_LEN (icmp_error_strings),
  .error_strings = icmp_error_strings,

  .n_next_nodes = 1,
  .next_nodes = {
    [ICMP_INPUT_NEXT_ERROR] = "error-punt",
  },
};
```

```
VLIB_REGISTER_NODE
(ip4_icmp_echo_request_node,static) = {
  .function = ip4_icmp_echo_request,
  .name = "ip4-icmp-echo-request",

  .vector_size = sizeof (u32),

  .format_trace = format_icmp_input_trace,

  .n_next_nodes = 1,
  .next_nodes = {
    [0] = "ip4-rewrite-local",
  },
};

VLIB_REGISTER_NODE (ip4_rewrite_local_node) = {
  .function = ip4_rewrite_local,
  .name = "ip4-rewrite-local",
  .vector_size = sizeof (u32),

  .sibling_of = "ip4-rewrite-transit",

  .format_trace = format_ip4_rewrite_trace,

  .n_next_nodes = 0,
};
```

```
CLIB_MULTIARCH_SELECT_FN
(vnet_interface_output_node_no_flatten);

VNET_DEVICE_CLASS (dpdk_device_class) = {
  .name = "dpdk",
  .tx_function = dpdk_interface_tx,
  .tx_function_n_errors = DPDK_TX_FUNC_N_ERROR,
  .tx_function_error_strings = dpdk_tx_func_error_strings,
  .format_device_name = format_dpdk_device_name,
  .format_device = format_dpdk_device,
  .format_tx_trace = format_dpdk_tx_dma_trace,
  .clear_counters = dpdk_clear_hw_interface_counters,
  .admin_up_down_function =
dpdk_interface_admin_up_down,
  .subif_add_del_function = dpdk_subif_add_del_function,
  .rx_redirect_to_node = dpdk_set_interface_next_node,
  .no_flatten_output_chains = 1,
  .name_renumber = dpdk_device_renumber,
};
```