Data center emulation and traffic management via a Cloud SDN Controller

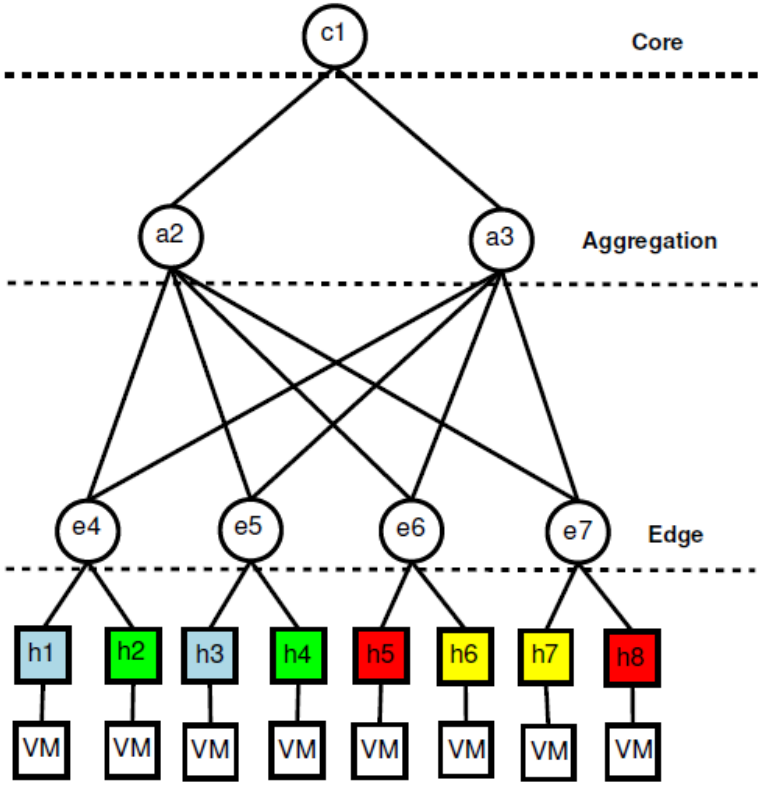
Emmanouil Lakiotakis

Course: HY436

Fall 2020

Definition

- Data center: a facility used to host computer systems and associated components, such as telecommunications and storage systems.
- In practice, a number of Virtual Machines hosted by distinct end-host servers. Servers are grouped based on their services creating **tenant groups** (different colors).



Data center requirements

- Efficient traffic routing across its expensive fabric
- Isolation
- Proper traffic engineering
- VM migration

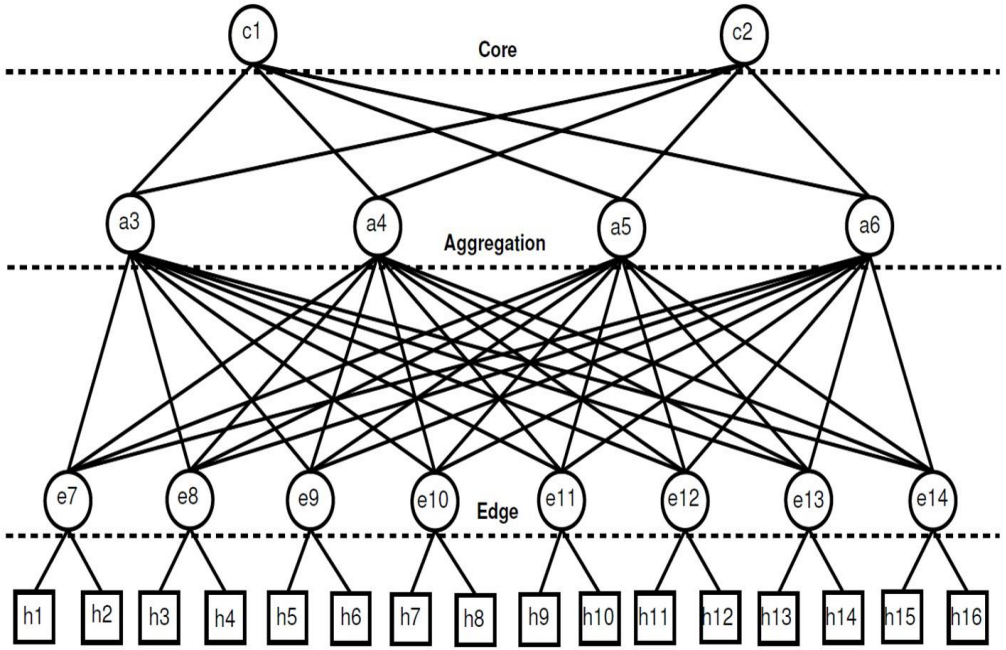
Assignment objectives

- Emulate a ‘sample’ data center topology via **Mininet**
- Satisfy conventional data center networking requirements using basic SDN principles via **POX**

Step 1: Network Topology

- Clos topology: tree-like with 3 layers
  - Core switches
  - Aggregation switches
  - Edge/access switches
- 2 parameters:
  - c: # core switches
  - f: (fanout) # children switches per parent
- **Follow depicted notation**

Example topology: c=2, f=2

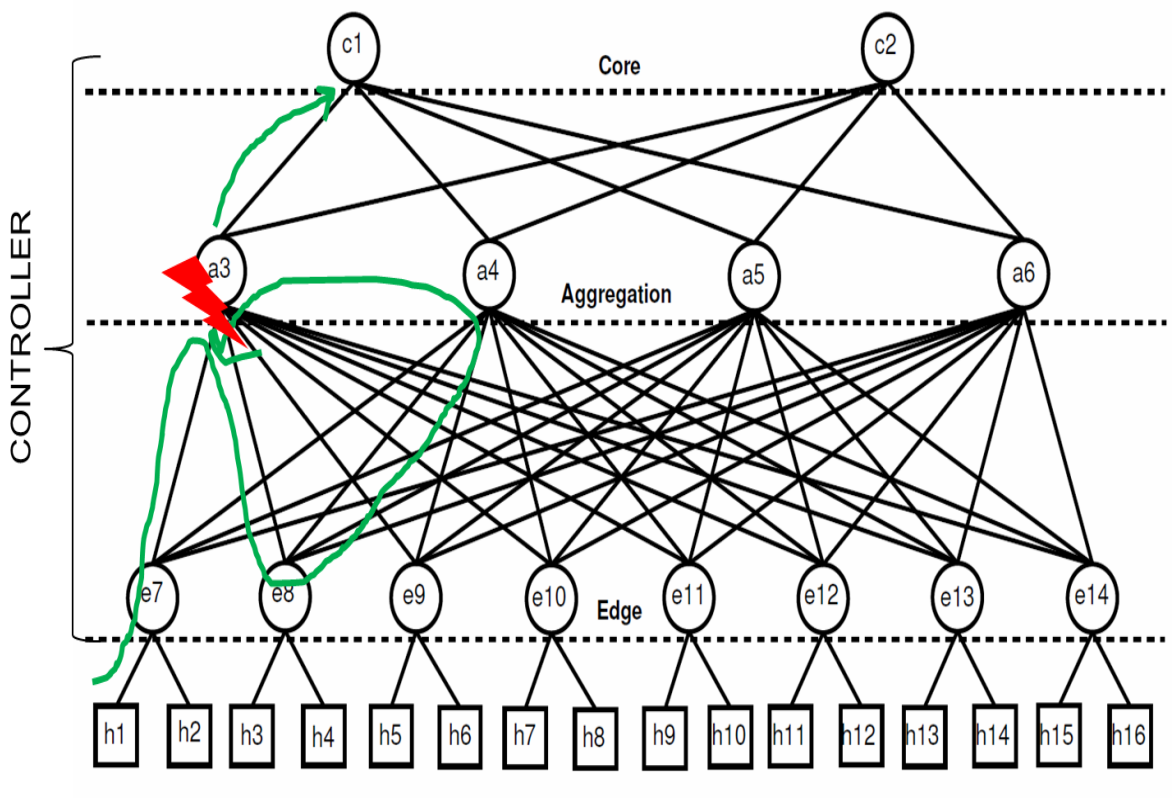


Step 2: Traffic Routing

- Based on shortest path routing between hosts
- Traffic classification/differentiation based on different protocols (similar to ToS matching)
- **Real traffic engineering objectives used in conventional data centers**

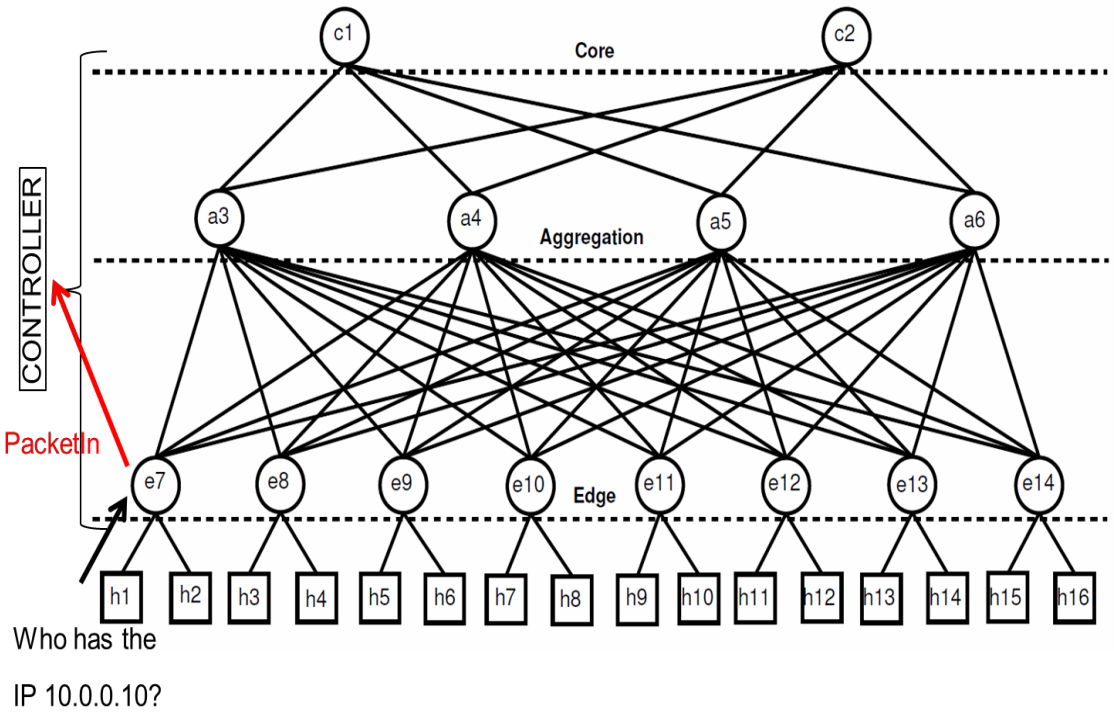
ARP packet handling

- In large scale topologies, conventional flooding approach can lead to routing loops
- Avoid ARP forwarding to higher switch levels
- **Solution: ARP proxying**



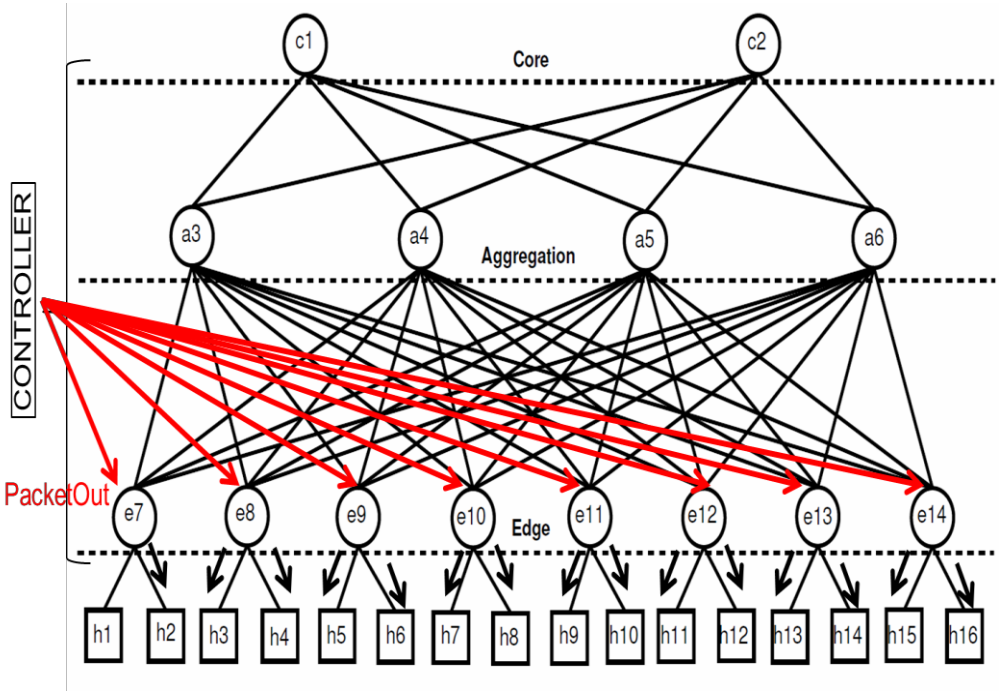
ARP proxying: ARP requests

- ARP requests raise PacketIn events
- 2 possible cases:
  - The Controller **does not know** how to answer to the ARP request
  - The controller **knows** how to answer the ARP request



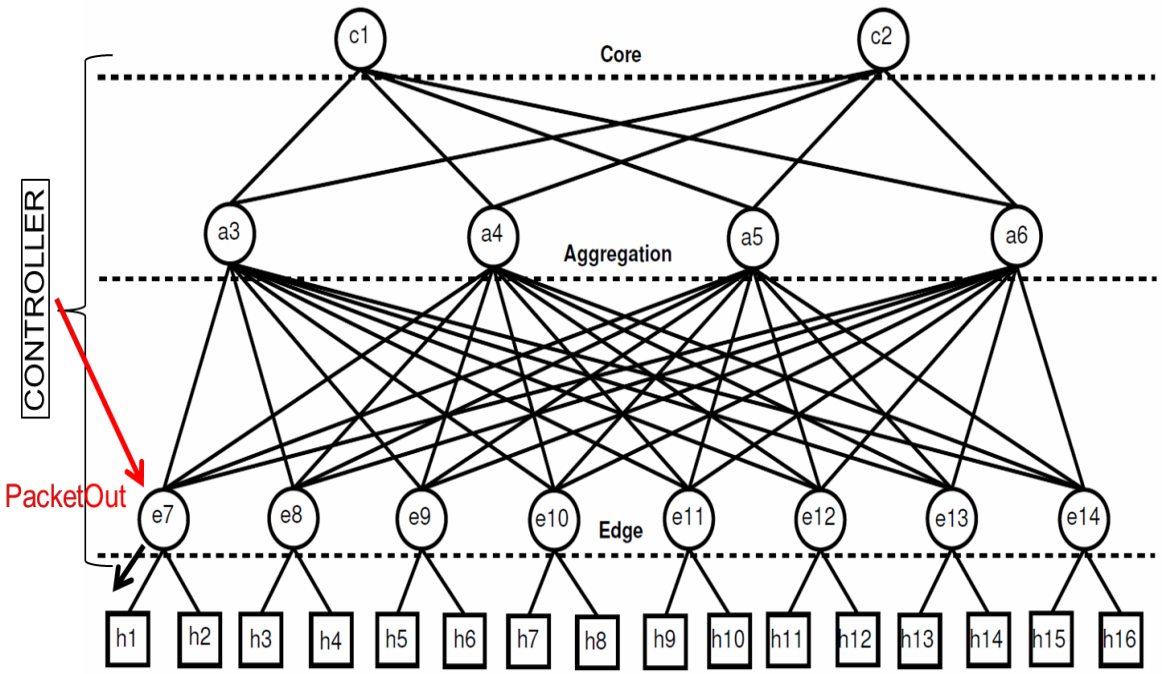
ARP proxying: ARP requests (a)

- When the controller **does not know how to answer to ARP request**:
  - 'Sophisticated' flooding only to ports that lead to hosts (flood_on_switch_edge())
  - Avoid routing loops
  - When host replies, forward the reply to the requesting host
- The controller knows which ports lead to switches and which ports lead to hosts



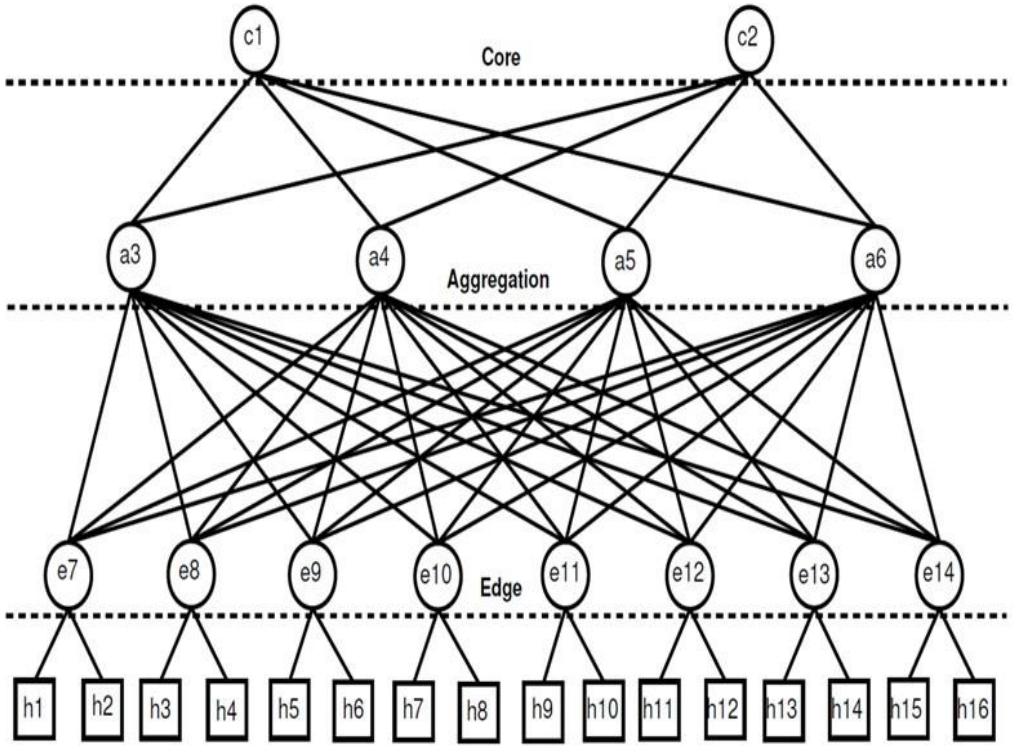
ARP proxying: ARP requests (b)

- When the controller **knows** how to answer to ARP request
- Acts as proxy creating the appropriate ARP reply and forwarding to the requesting host



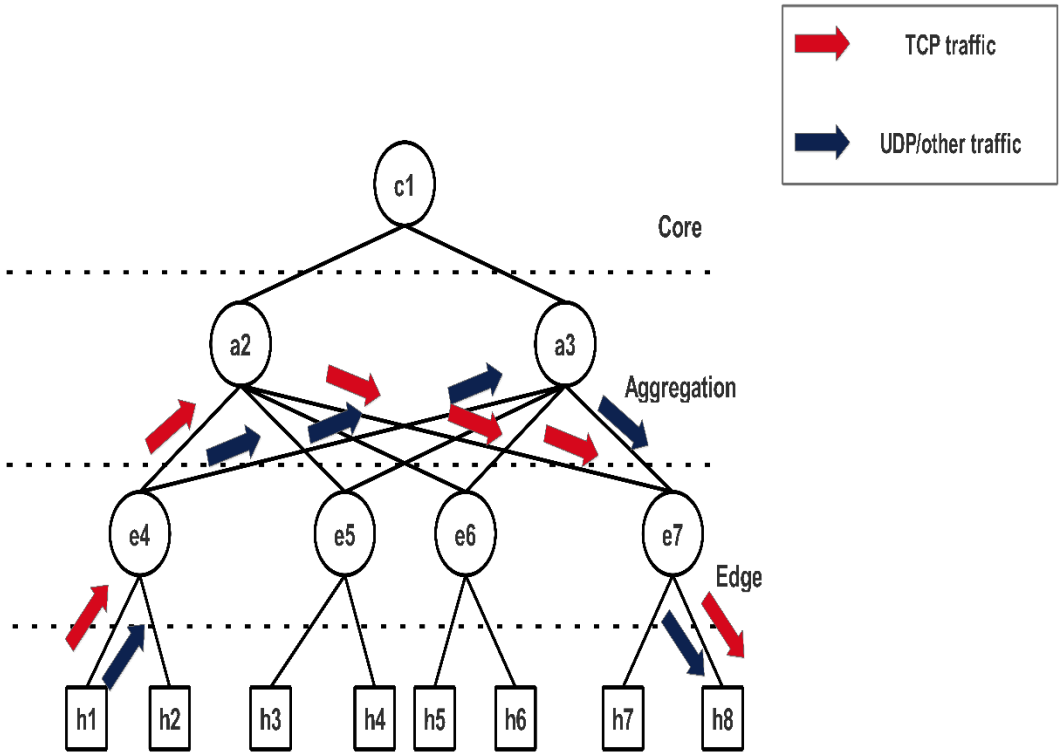
Shortest Path routing

- Use NetworkX Python package
  - Data center graph representation (G)
  - Evaluate all shortest paths in terms of #hops between all switches (Hint: all_shortest_paths(G, source, target))
  - All this functionality is described in SwitchWithPaths class



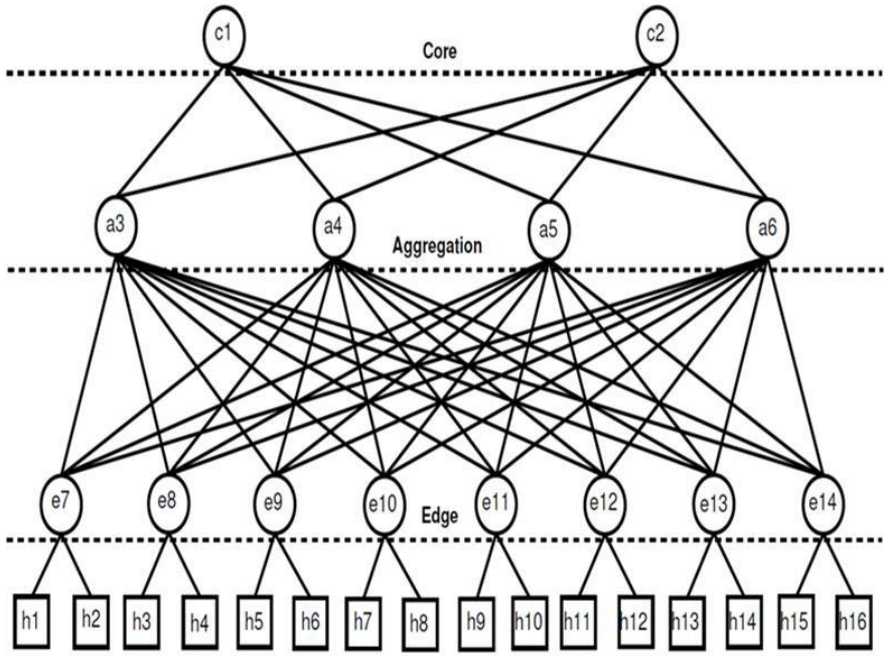
Traffic Differentiation

- Different type of traffic (TCP vs UDP/other) follow different paths
- **Hint: function getPathsperProto ()**



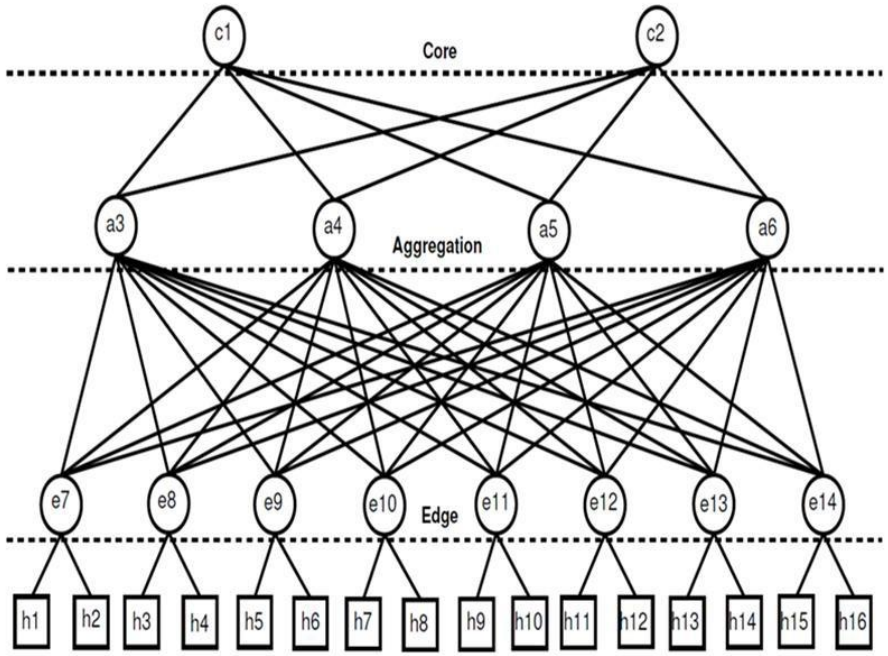
Flow rules installation

- Flow rule installation is described in install_end_to_end_IP_path function
- Called in PacketIn's case
- 2 cases:
  - If Event switch==dest switch=> install flow rule + packet out
  - Otherwise => path evaluation



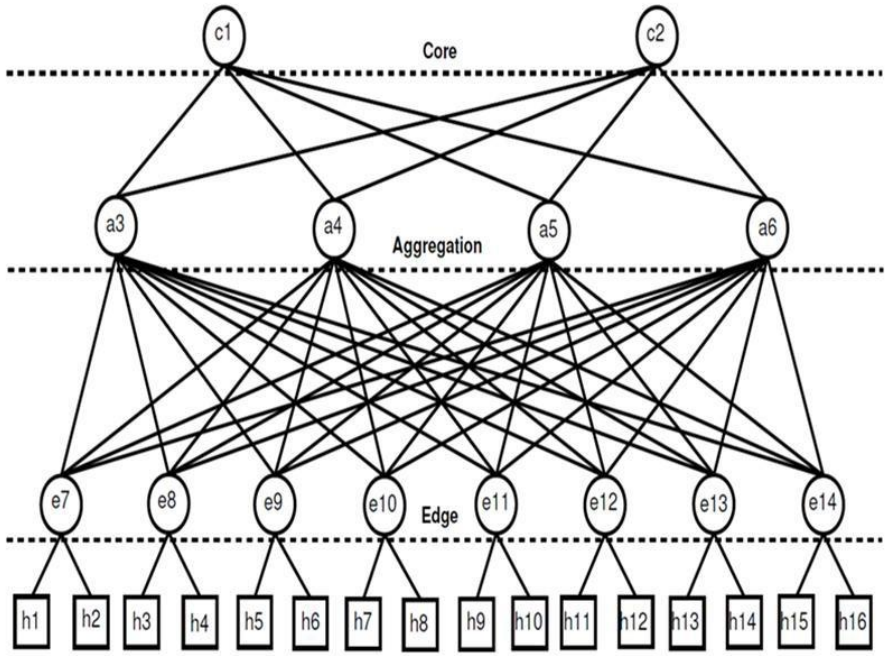
Flow rules installation

- Topology discovery via POX's Discovery component
- After path evaluation, choose the appropriate path
- Path installation in inverse fashion but in straight direction
- Example: Path: A->B->C->D
  1. C->D
  2. B->C
  3. A->B



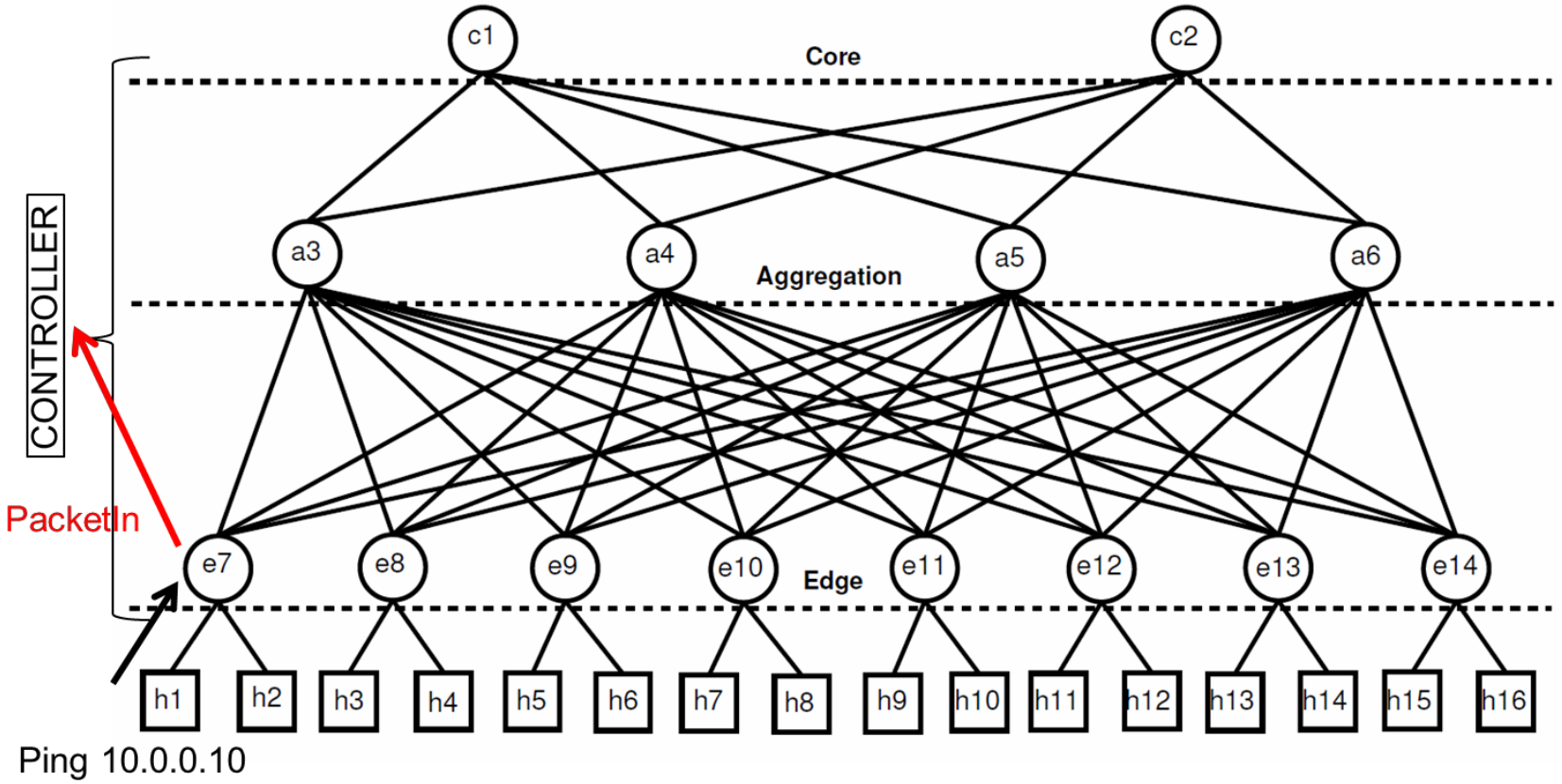
Flow rules installation

- Flow rule installation via install_output_flow_rule function
- Entire packet forwarding (no buffer id used)
- Information about ports interconnection between switches in sw_sw_ports variable



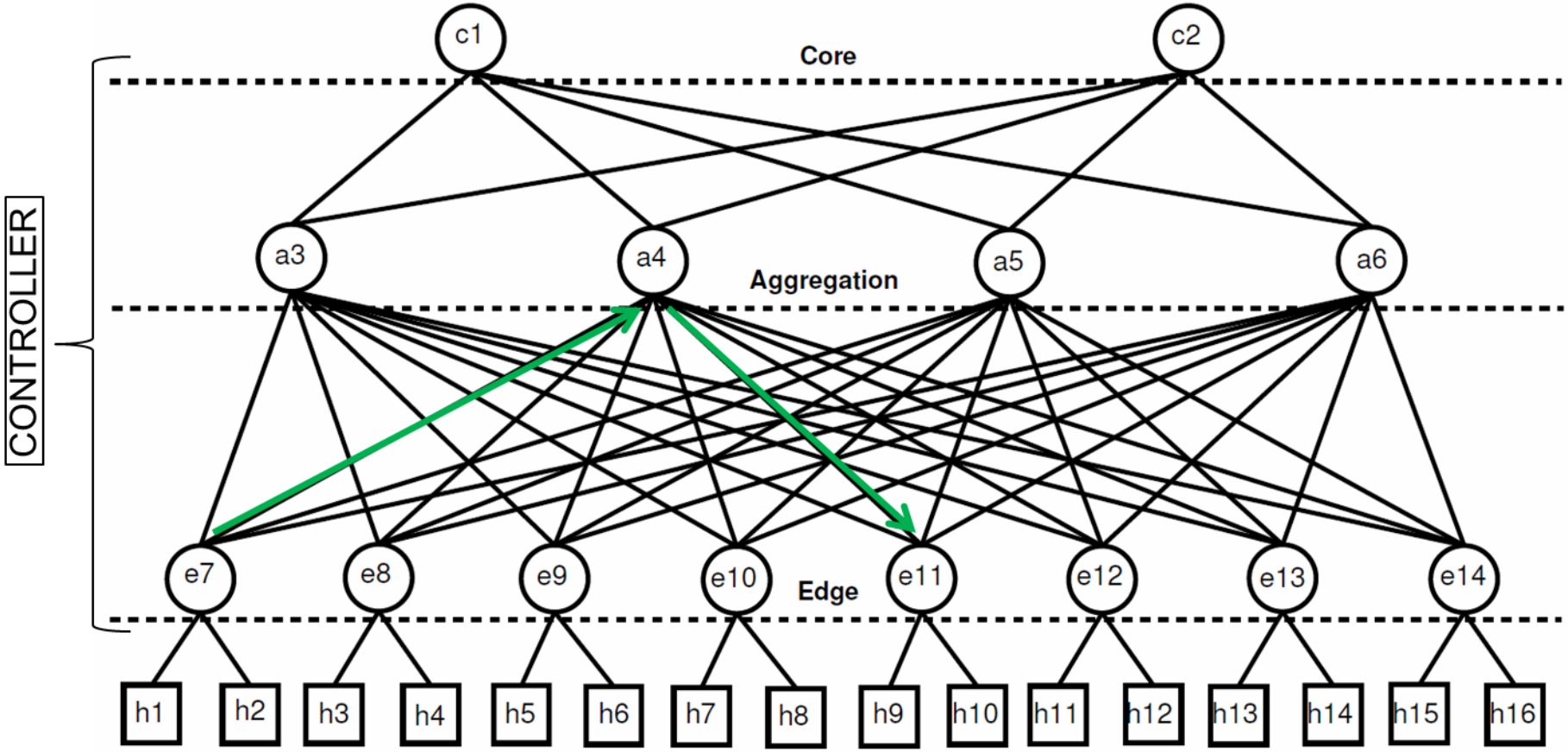
Flow rules installation - Example

- IP PacketIn



Flow rules installation - Example

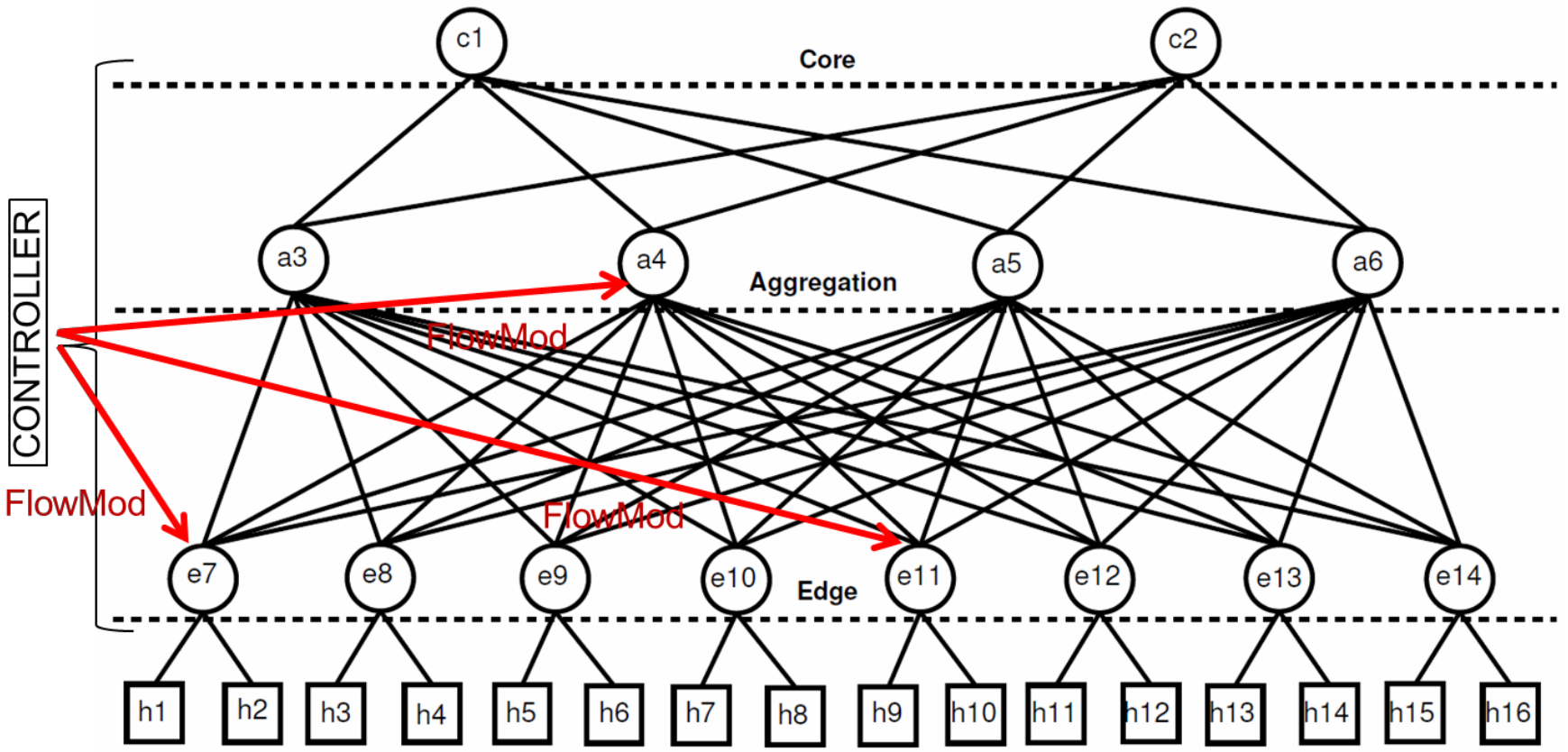
- Path Evaluation



➔ Now the controller chooses a random shortest path between e7 and e11!

Flow rules installation - Example

- Path Setup

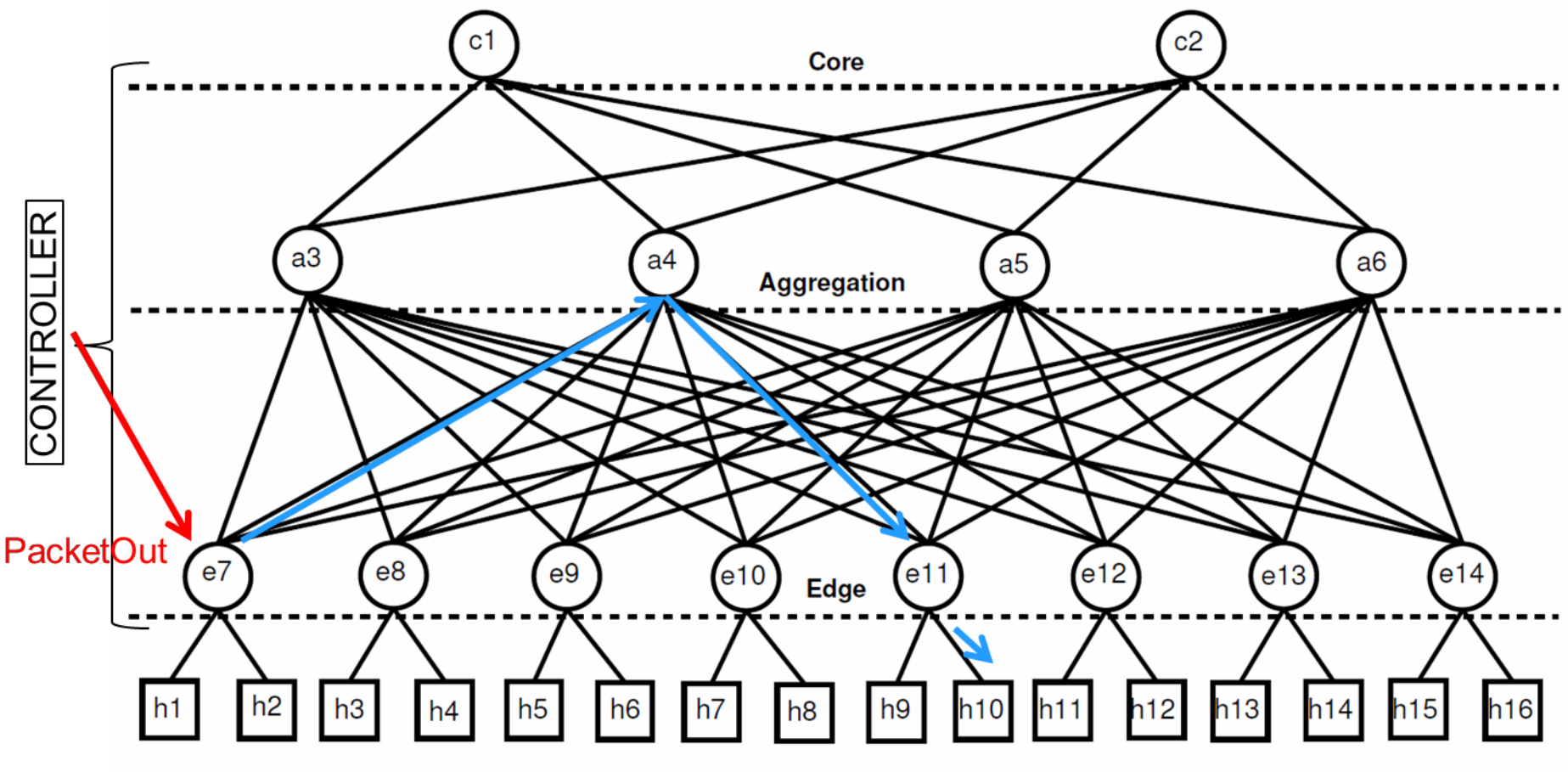


➔Side-note: install flow mods in reverse to avoid even more PacketIns!

➔What are potential timing issues here? Can we tackle them?

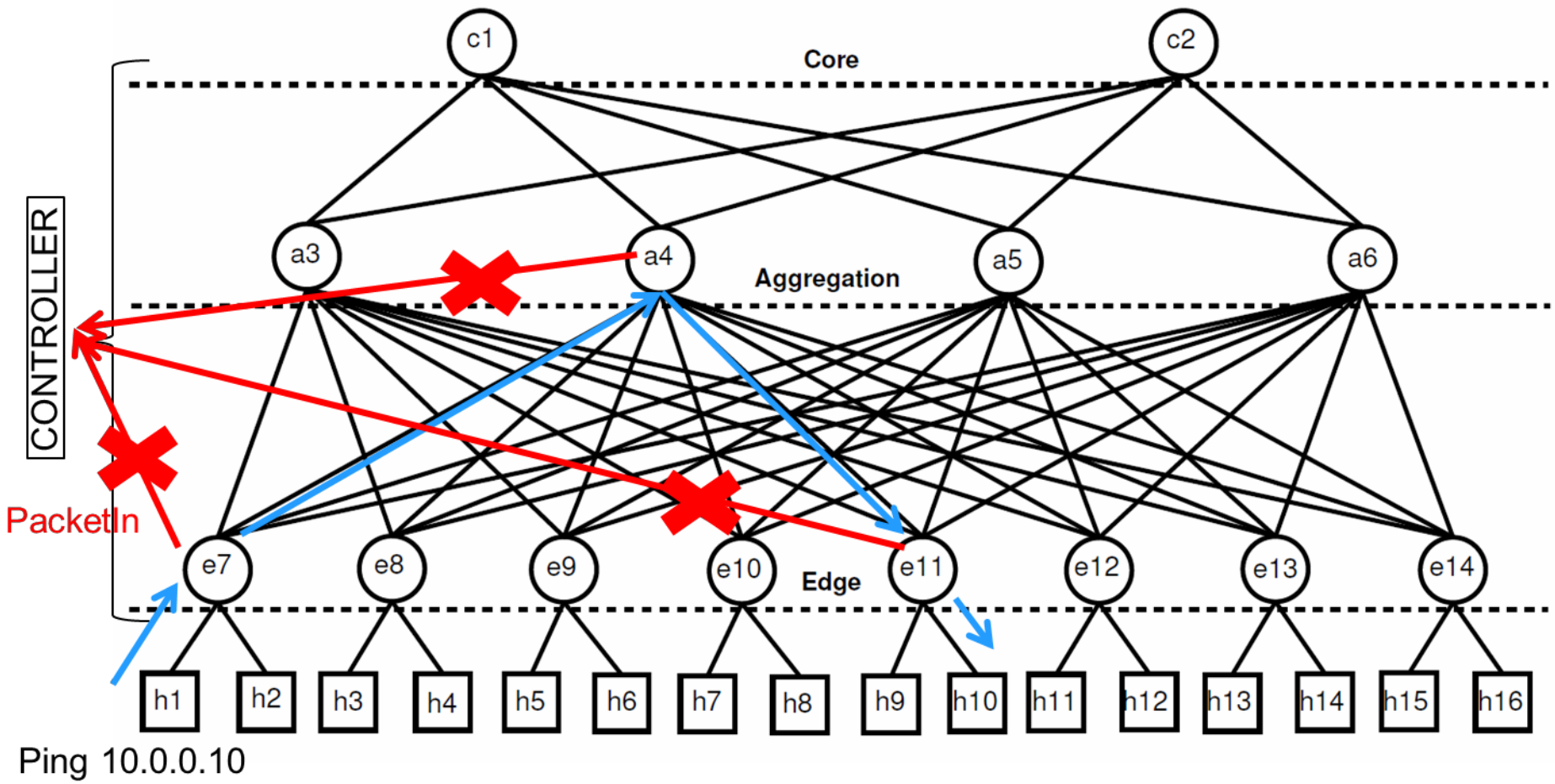
Flow rules installation - Example

- **Current Packet** Forwarding



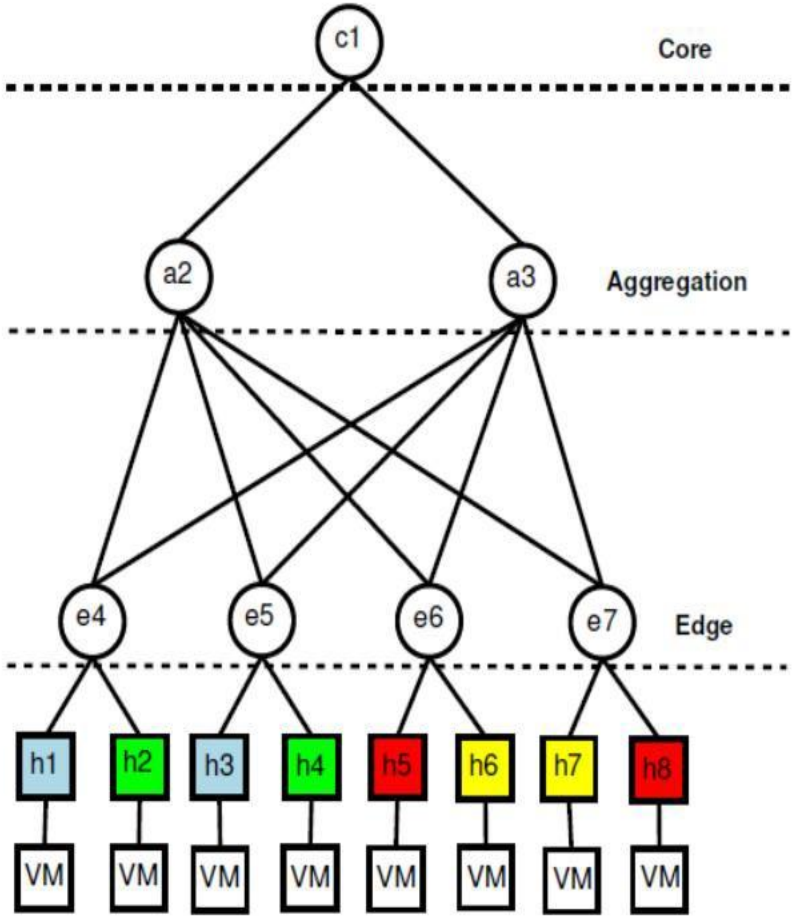
Flow rules installation - Example

- **Coming Packets** Forwarding



Step 3: Firewall/Isolation

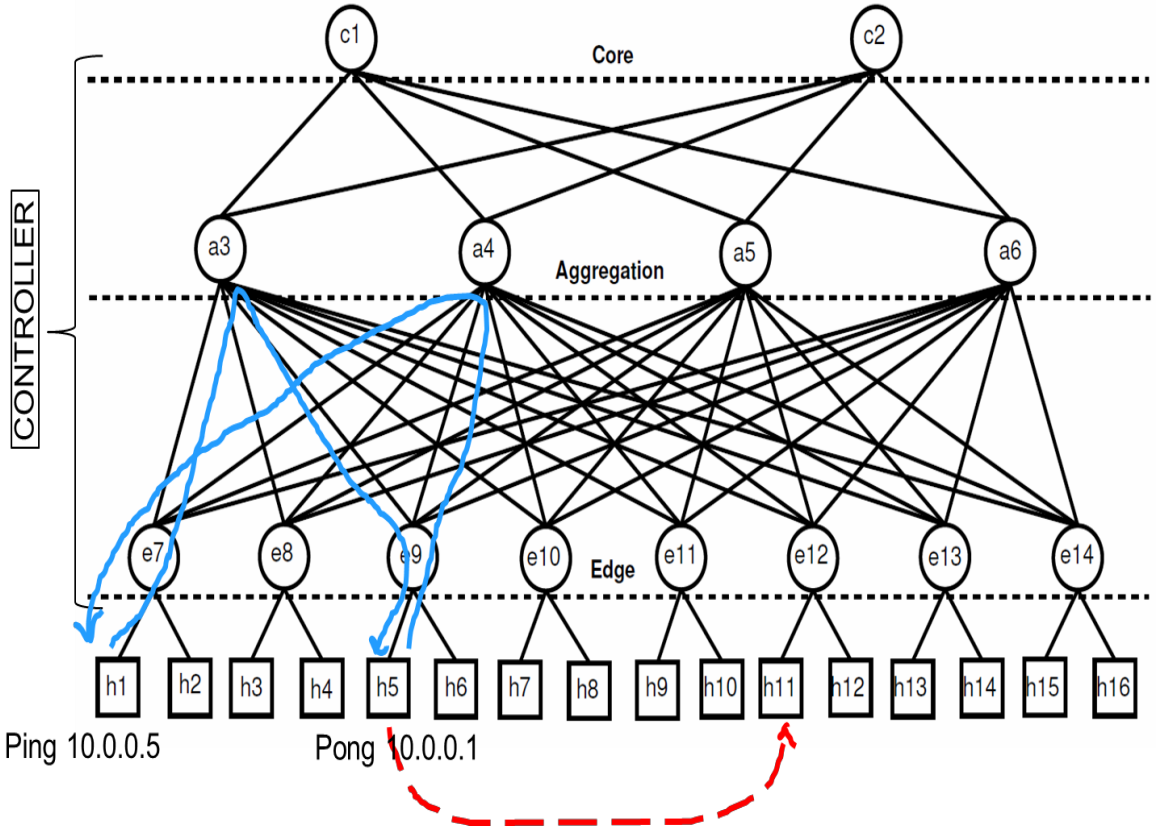
- Communication is allowed between hosts that belong to the same tenant group(same color)
- Firewall policies described in firewall_policies.csv
- Policies refer to **both IP and ARP traffic**



Step 4: Migration/request

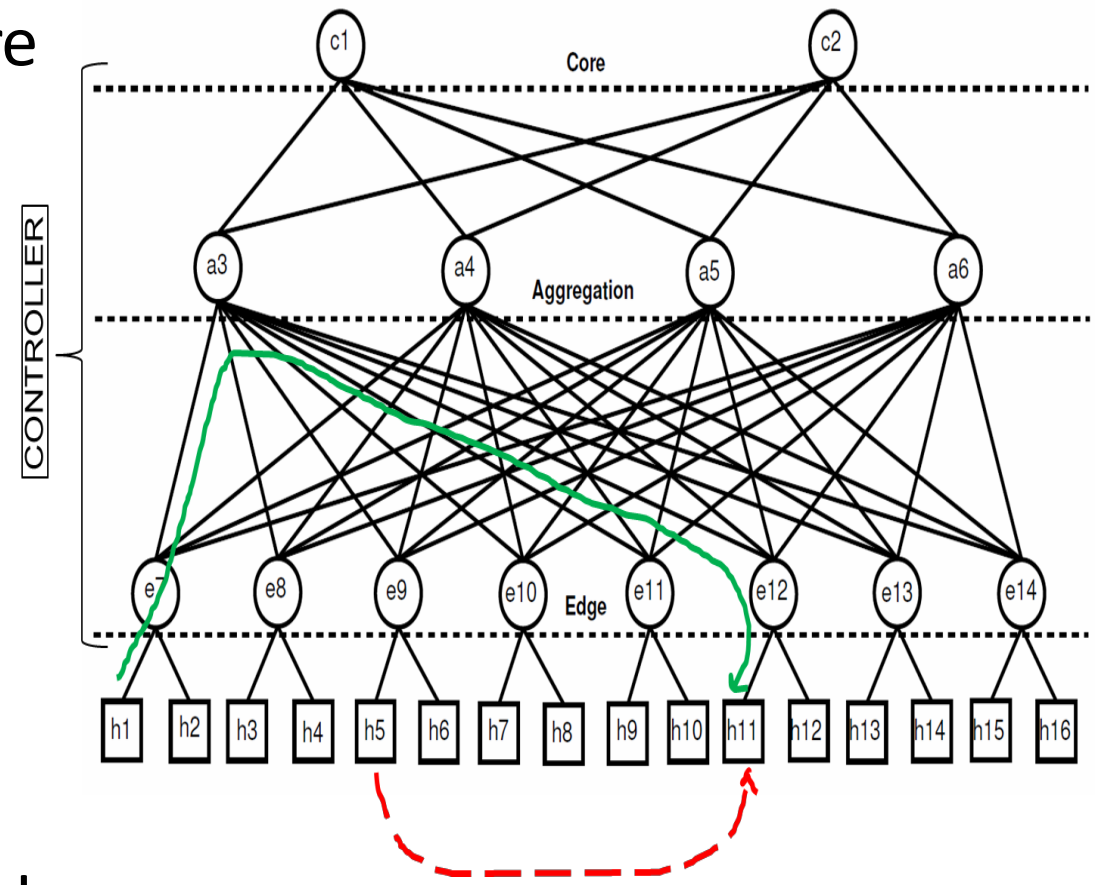
- Migration reason:
  - Maintenance
  - Failure
  - Resource allocation optimization
- Objective: **Transparency in IP level**
- Migration event described in migration_events.csv

Example: h1 ping h5 which is migrated to h11



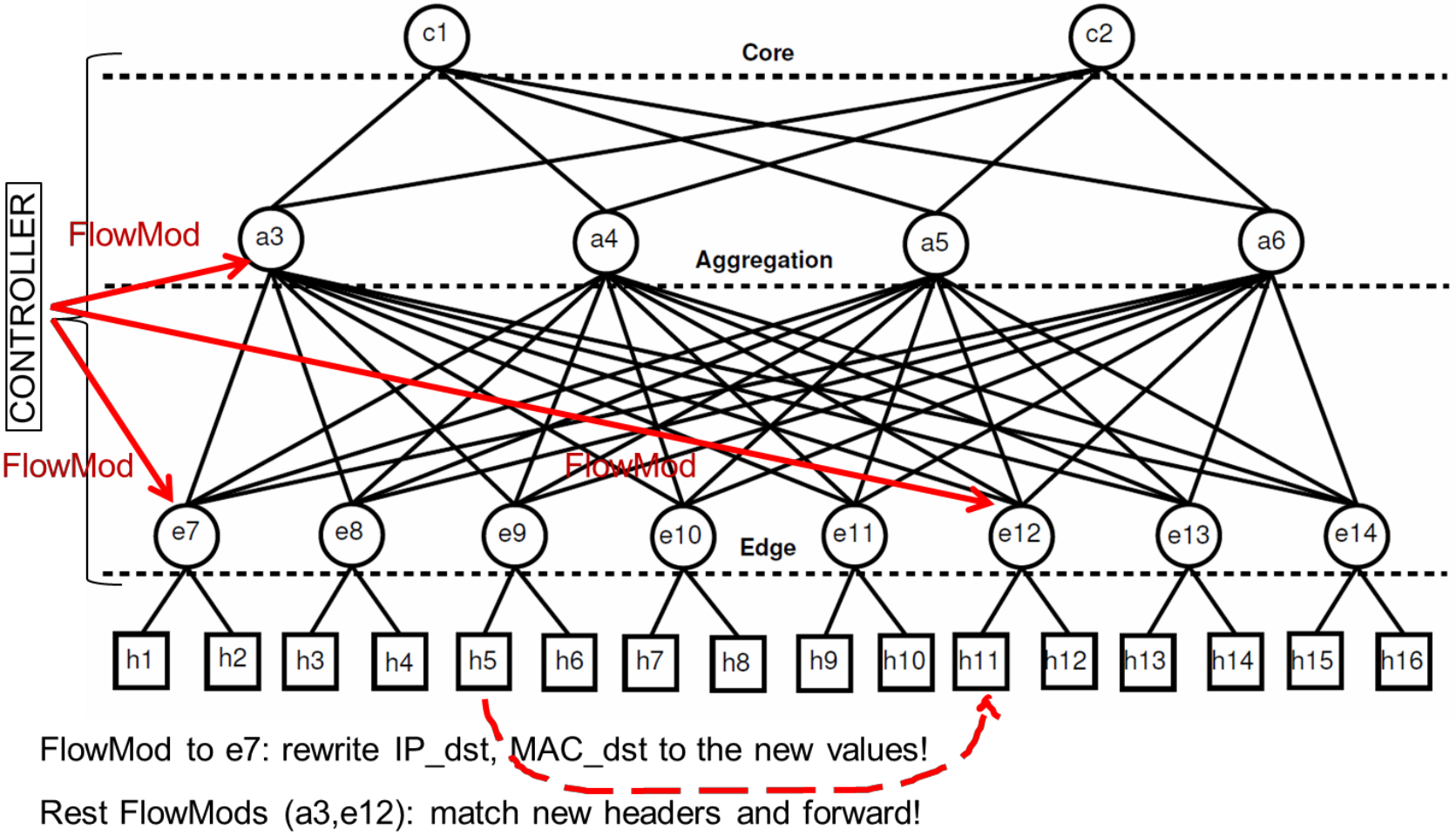
Step 4: Migration/request

- After migration, all installed flow rules are removed
- New PacketIn events are raised by packets destined to migrated host
- Path evaluation towards h11
- Destination IP and MAC re-write required



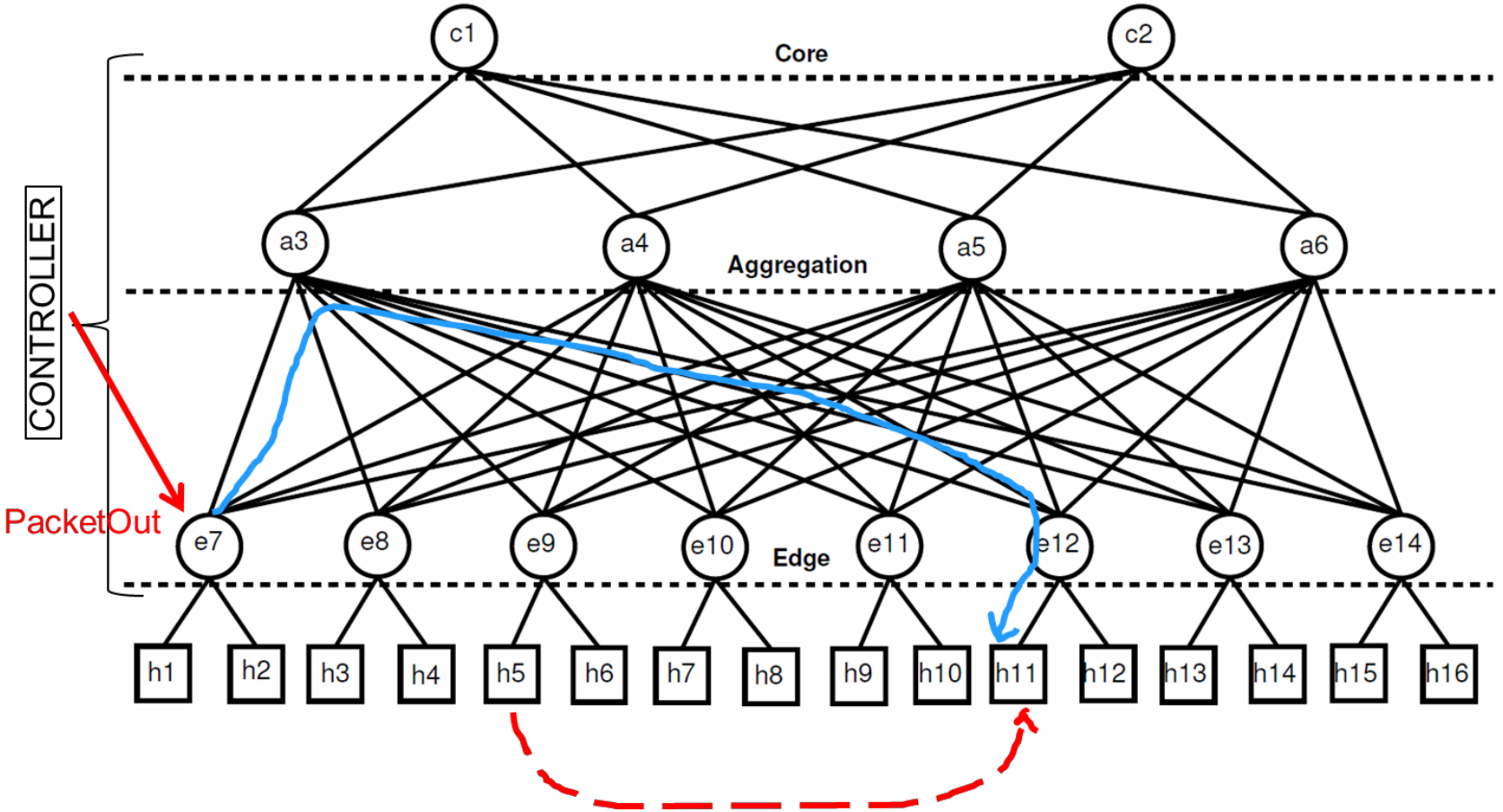
Step 4: Migration/request

- New flow rules installation



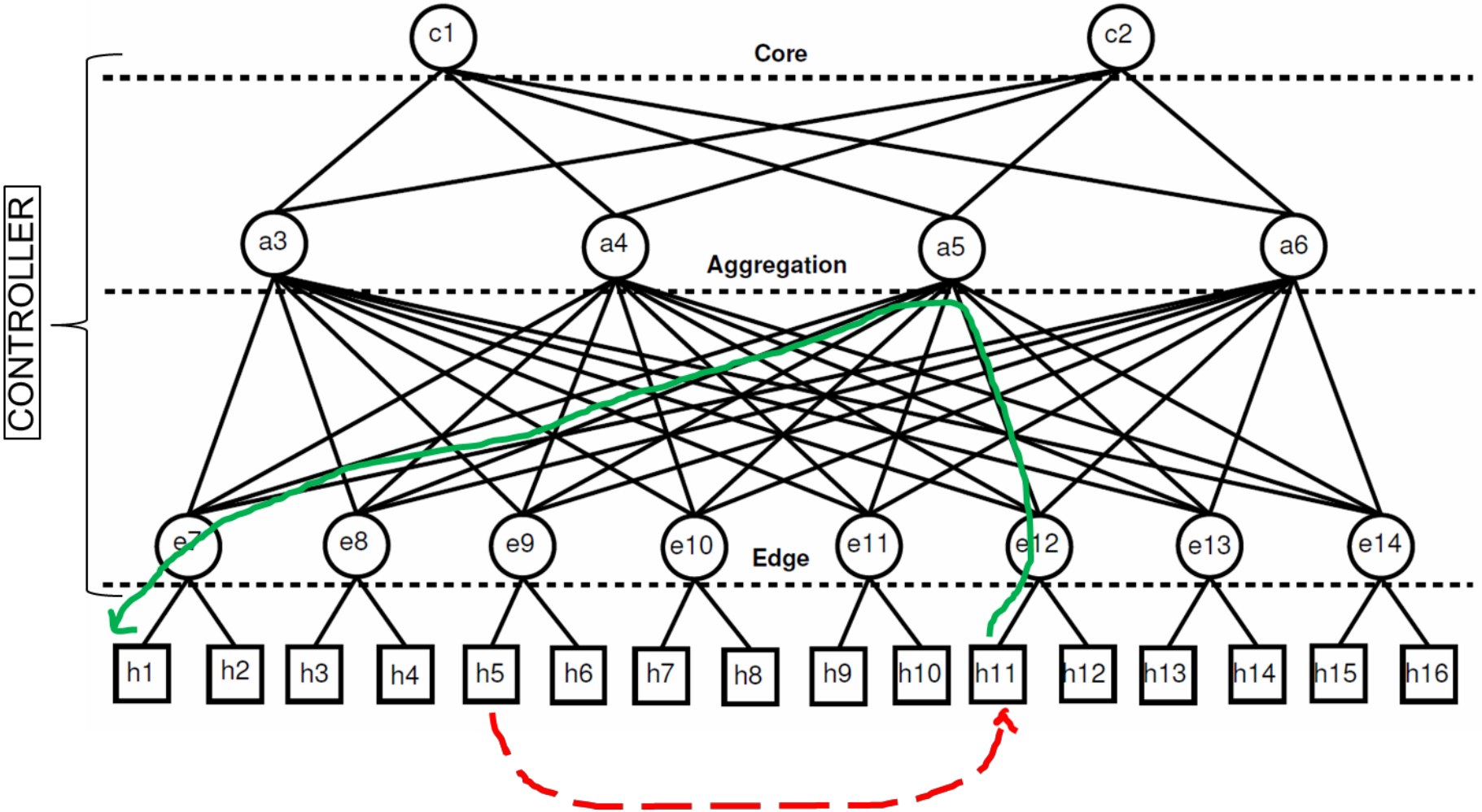
Step 4: Migration/request

- Packet processing/forwarding



Step 4: Migration/reply

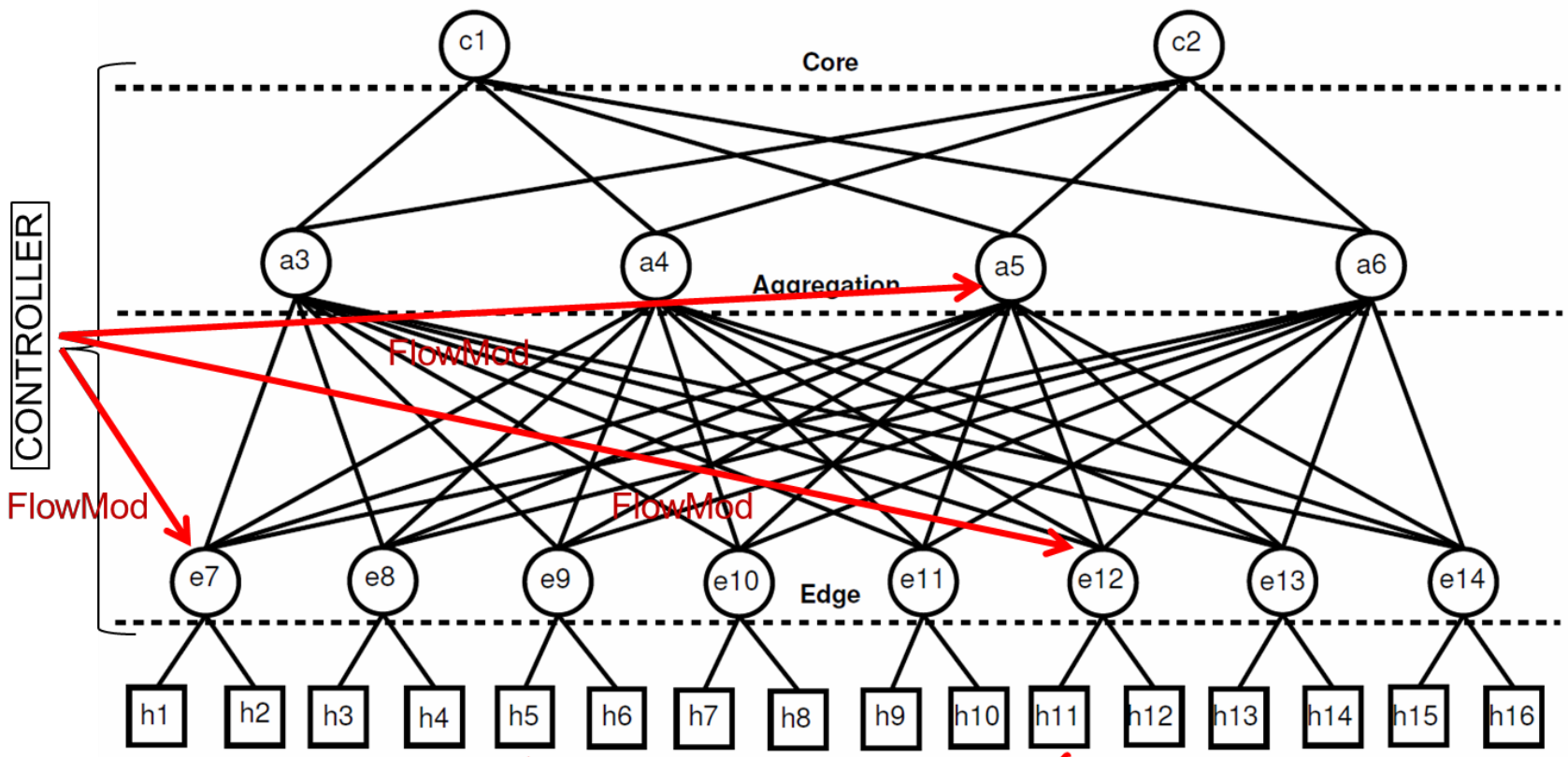
- Reply Path evaluation



Step 4: Migration/reply

- Reply Path Setup

Source IP and MAC re-write required

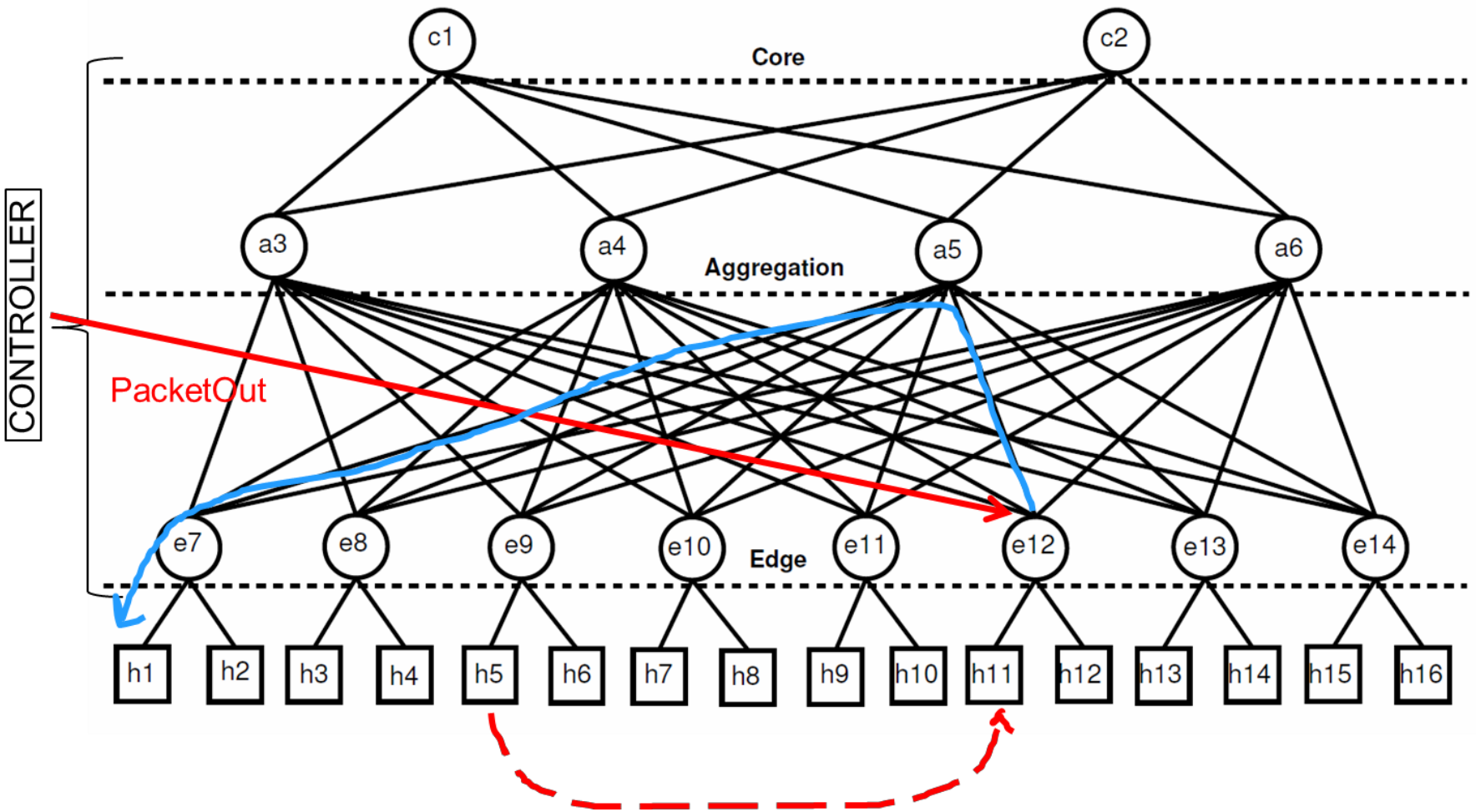


FlowMod to e12: rewrite IP_src, MAC_src to the old values!

Rest FlowMods (a5,e7): match old headers and forward!

Step 4: Migration/reply

- Reply processing/forwarding



General guidelines

- Write your code in
  - clos_topo.py
  - CloudNetController.py
- Read carefully the assignment description
- For your questions, use Moodle forum
- 3 exercise sessions devoted to assignment 2
- **Good luck**


Спасибо
Gracias
شكرا
Obrigado
Спасибо
Dank U
Grazie
Ευχαριστώ
Danke
Merci
Obrigado
Diolch
Thank You
Ngiyabonga
Dank U
Thank
You
Merci
Tack
Dzięki
Ευχαριστώ
Danke
Grazie
תודה
Dank U
Diolch
Grazie
Tack
Ευχαριστώ
Terima Kasih
Merci
Diolch
Gracias
תודה