CONIA: Content (Provider)-Oriented Namespace-Independent Architecture for Multimedia Information Delivery

*Eman Ramadan, **Arvind Narayanan**, Zhi-Li Zhang*

Department of Computer Science & Engineering

University of Minnesota, Minneapolis, USA

July 3, 2015

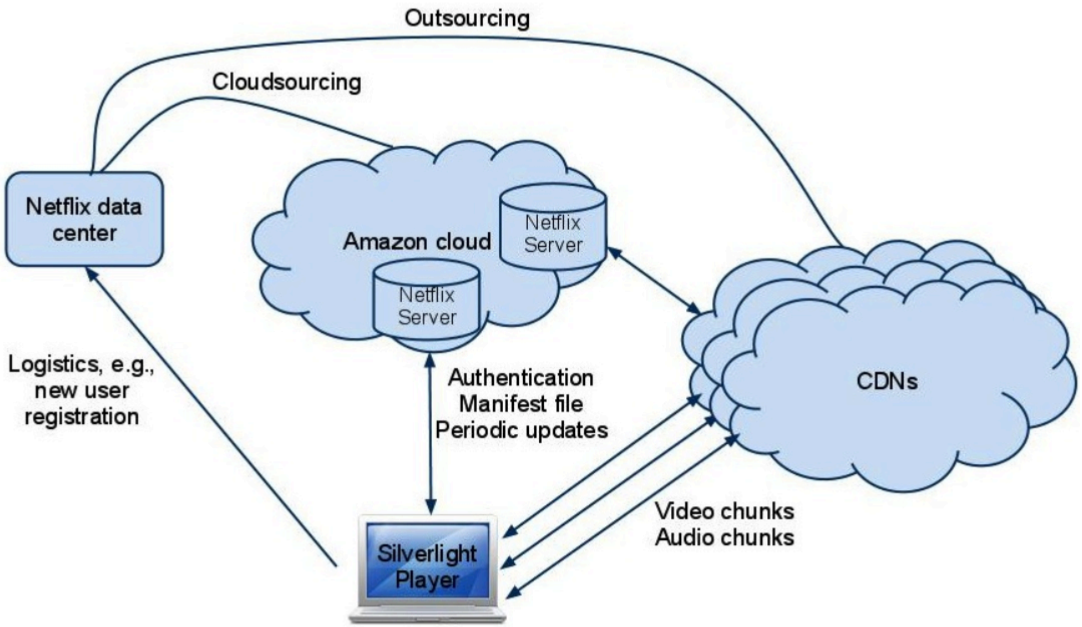
MuSIC (co-located with ICME) 2015, Torino, Italy

Outline

- Introduction & Motivation
- Architecture
- Hints on Design & Implementation
- Use Cases
- Conclusion

Limitation of Today's Internet

- Streaming services have expanded the role of Content Distribution Networks (CDNs) to a new level
- For example, Netflix consumed almost a third of North America's downstream traffic in 2014[1]



Netflix Architecture [2]

Netflix designed *OpenConnect* to have more control in content distribution [3]

Future Internet Architecture Design Requirements

- Better handling the ***diversity*** and ***complexity*** associated with ***multimedia*** content
  - For example, video object is composite
  - no single *namespace* can fit it all

- Need for ***Content providers (CPs)*** to have a larger say in **provisioning** and *dynamically* **distributing** content
  - e.g., deploy their *own* load balancing/cache management policies

- Take into account the ***network economics*** of content delivery to make the ICN design **economically viable**

CONIA: A New ICN Architecture

- Two basic tenets underlying all ICN designs:
  1. Content is the **first-class object**
  2. Content storage should be ***part*** of the **network substrate**

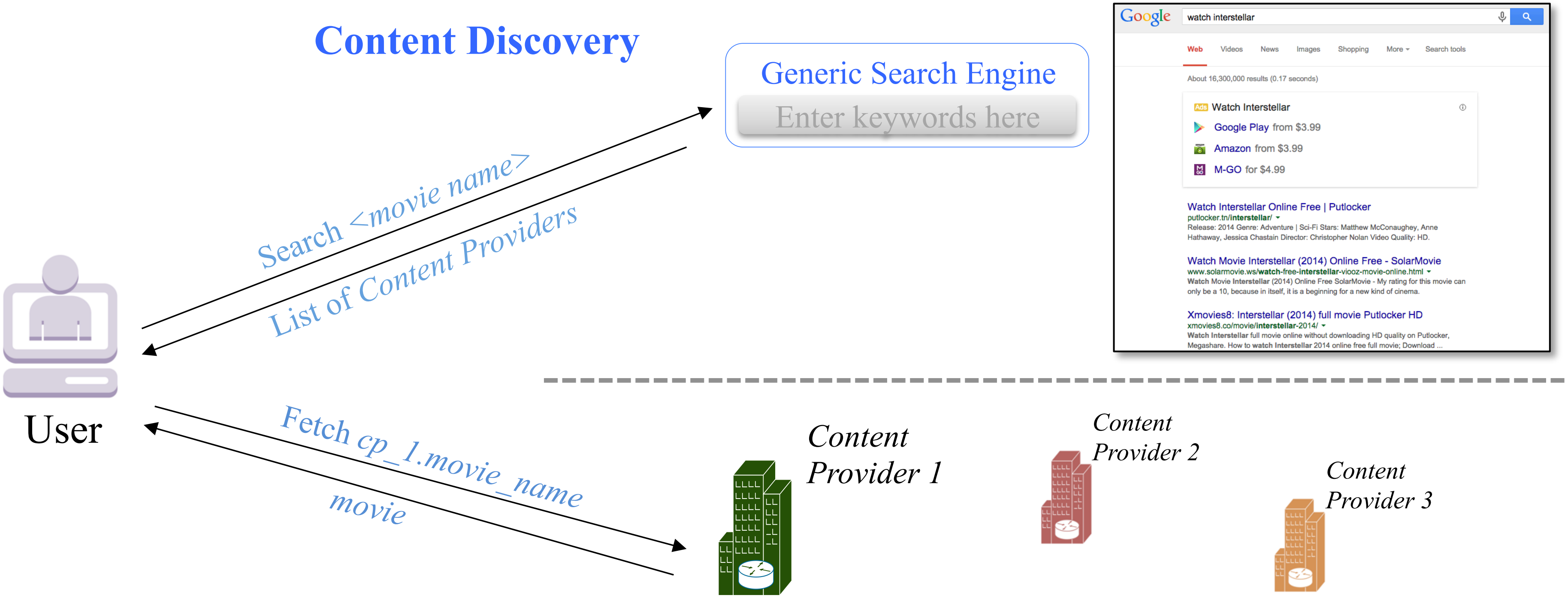
- **But**, we put forth a third tenet:
  3. One must not *dictate* how the namespace for content is designed for *all* content *providers*
     - **so as to enable a scalable, robust, economically viable, and evolvable ICN architecture**

Two Dimensions of CONIA

Content Discovery - Mapping user's search query to actual content name

Content Delivery - The process of requesting and delivering content

We separate the two …



Google | watch interstellar

Web | Videos | News | Images | Shopping | More ▾ | Search tools

About 16,300,000 results (0.17 seconds)

Ads Watch Interstellar

Google Play from $3.99

Amazon from $3.99

M-GO for $4.99

Watch Interstellar Online Free | Putlocker
putlocker.tn/**interstellar**/ ▾
Release: 2014 Genre: Adventure | Sci-Fi Stars: Matthew McConaughey, Anne Hathaway, Jessica Chastain Director: Christopher Nolan Video Quality: HD.

Watch Movie Interstellar (2014) Online Free - SolarMovie
www.solarmovie.ws/**watch-free-interstellar**-viooz-movie-online.html ▾
Watch Movie **Interstellar** (2014) Online Free SolarMovie - My rating for this movie can only be a 10, because in itself, it is a beginning for a new kind of cinema.

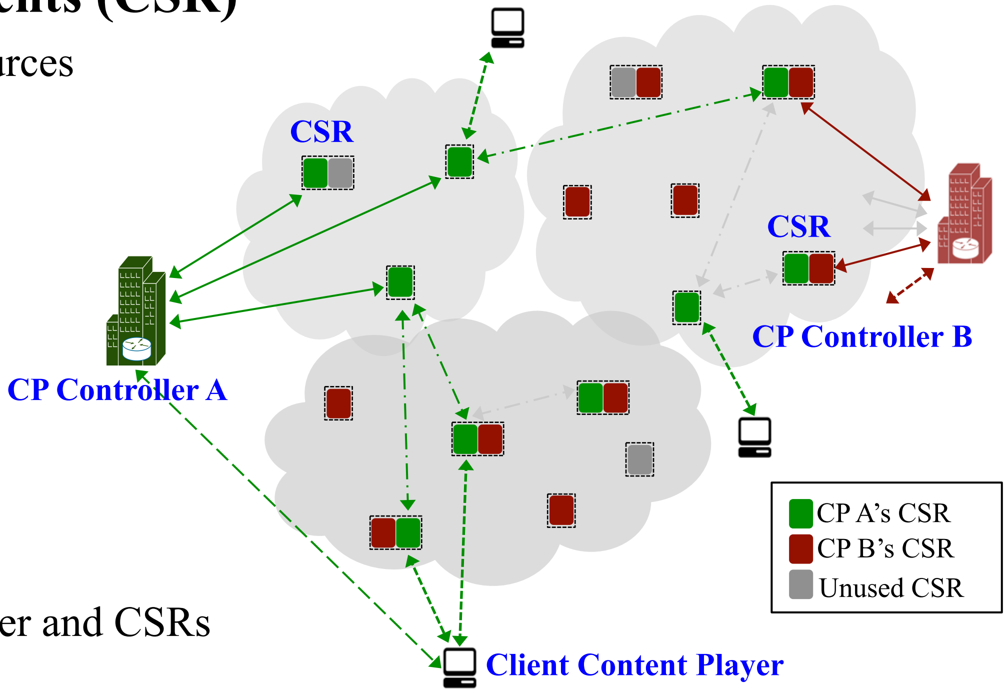
Xmovies8: Interstellar (2014) full movie Putlocker HD
xmovies8.co/movie/**interstellar**-2014/ ▾
Watch Interstellar full movie online without downloading HD quality on Putlocker, Megashare. How to **watch Interstellar** 2014 online free full movie; Download ...

Content Delivery

CONIA: Content Delivery Architecture

✧ Three key components

- **Content Store and Routing elements (CSR)**
  - Generic, programmable, and shared resources
  - Offered by third party entities
- **CP Controller**
  - Content-provider (CP) specific
  - Provision & manage CSRs
- **Client Content Player**
  - Generic or CP-specific software
  - Allows users to interact with CP Controller and CSRs



🟩	CP A’s CSR
🟥	CP B’s CSR
⬜	Unused CSR

✧ An ***open*** and ***standardized*** control framework API used for interaction between the components.

Content Store and Routing (CSR)

Functions

- *Shared* resource
- *Caches content; routes* requests and data
- Provides *basic functions* required for *resource management* & *content delivery*
- Stores *CP-specific control logic* specifying how to handle requests & data

CSR → CP Controller Interactions

- Reports statistics

CSR → CSR Interactions

- Reports health information
- Content offloading

CSR → Client Content Player Interactions

- Responds to client requests with *data*

CP Controller

Functions

- *Provisions* CSRs
- Defines its own *namespace*
- Decides “*what to cache*”, “*where to cache*”
- Defines the *control logic* used to handle and forward requests and data
- *Maps* client requests to CSRs

CP Controller → CSR Interactions

- Pushes the namespace and content to CSRs
- Installs the control logic into CSRs

CP Controller → Client Content Player Interactions

- Responds to client requests with *content map* (such as MPD)
- *Dynamically* generates content map using the global view and the collected statistics

Client Content Player

Functions

- *Interprets* the Content Map
- *Renders* and *displays* the *content*

  e.g., web browser, video player

Client Content Player → CP Controller Interactions

- Sends user request to CP controller and gets *Content Map* in return
- Reports statistics

Client Content Player → CSR Interactions

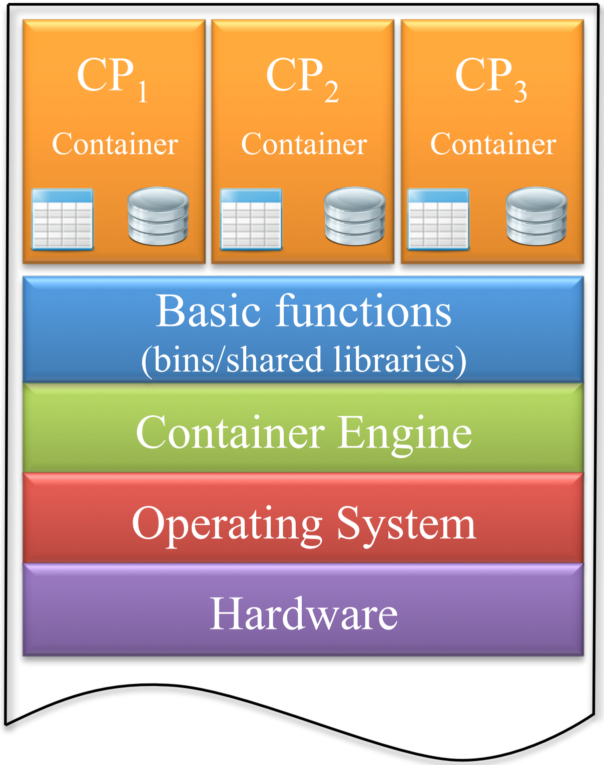
- Fetches content

CSR Design and Implementation

Current Research

- A programmable "**open**" box

- A logical view of CSR



- Basic *shared* functions/libraries/services used by all the containers

  e.g., web server, socket functions, I/O functions, etc.

- Every Content Provider (CP) has *full control* over its container

- Every container has
  - a *storage* device used to cache content, meta-data, statistics, etc.
  - a *Content Control Logic Table (CCLT)* used to control the content delivery plan

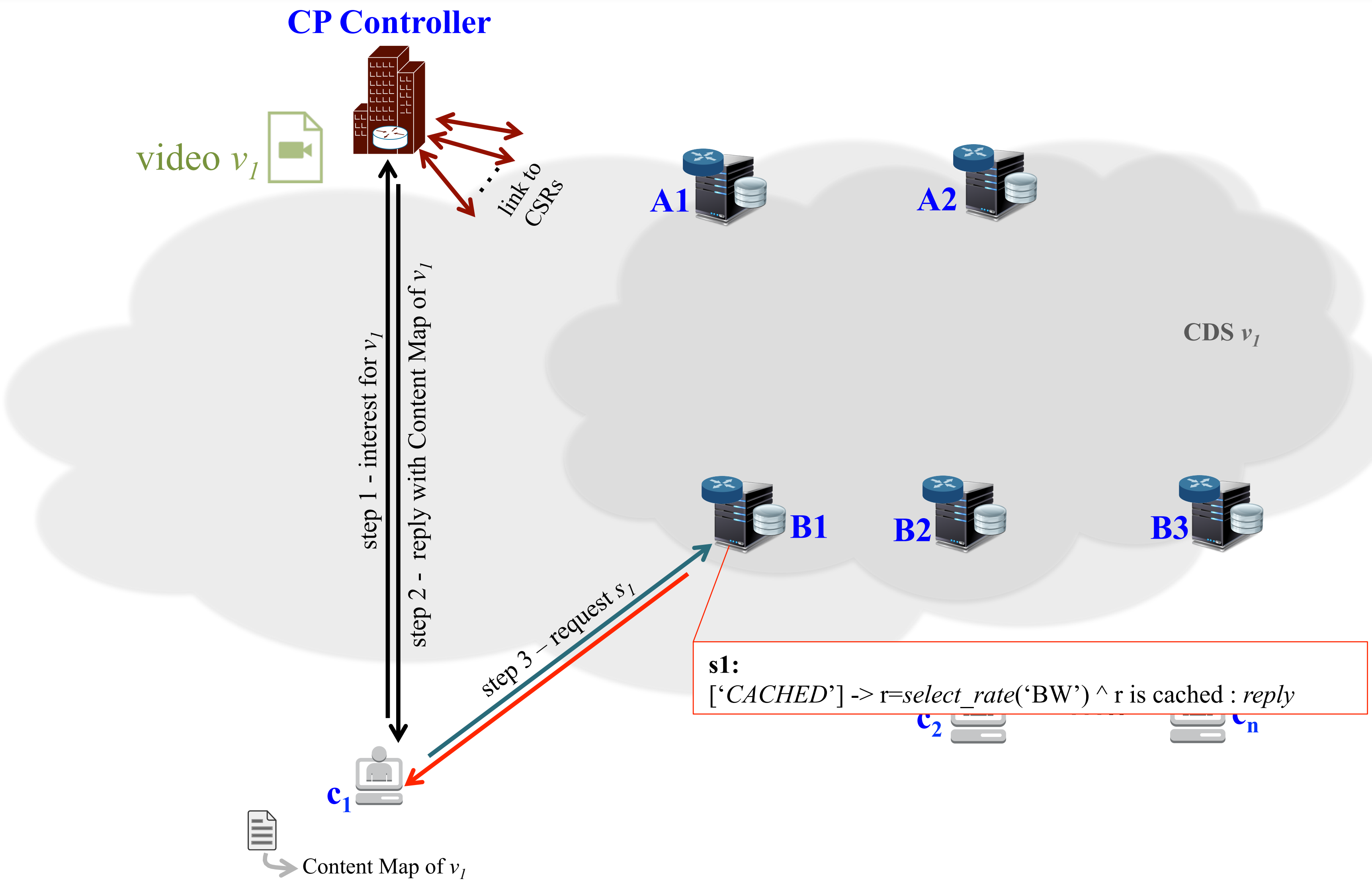
CCLT Design

Object ID	Statistics	Control Logic
…	…	…

Content Control Logic Table (CCLT) Structure

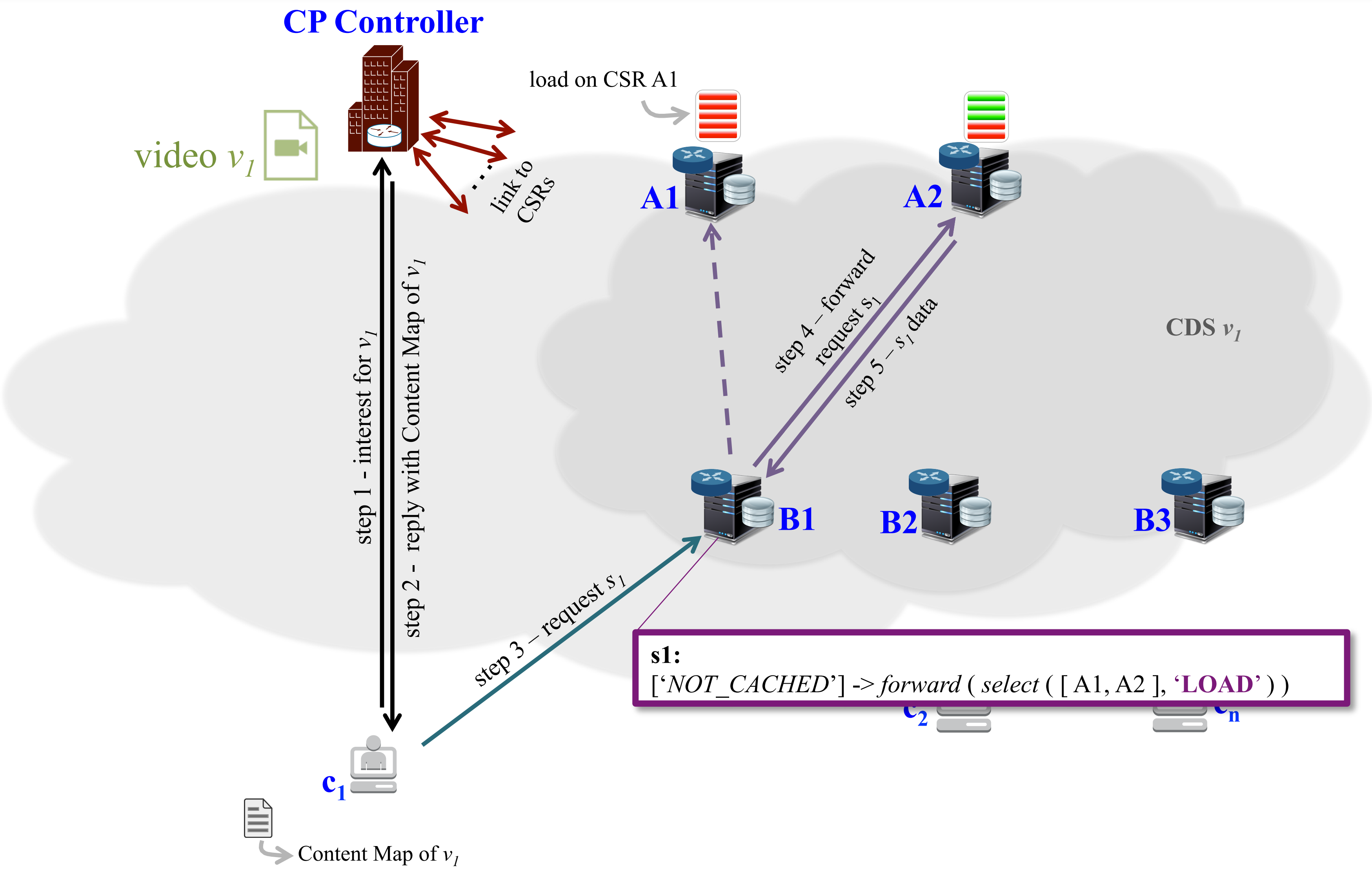
- An entry in CCLT is composed of three fields
  - object ID – content name or identifier *(granularity of object decided by CP)*
  - statistics – counters, etc.
  - control logic – how to handle the object
- The control logic in CCLT is expressed using a *declarative language* and defined in terms of two categories of *context*
  - Content-related context – the state of an object *e.g., cached or not-cached, etc.*
  - System-related context – the state of CSR/network condition *e.g., load on CSR, etc.*

Use Cases: A Simple Example



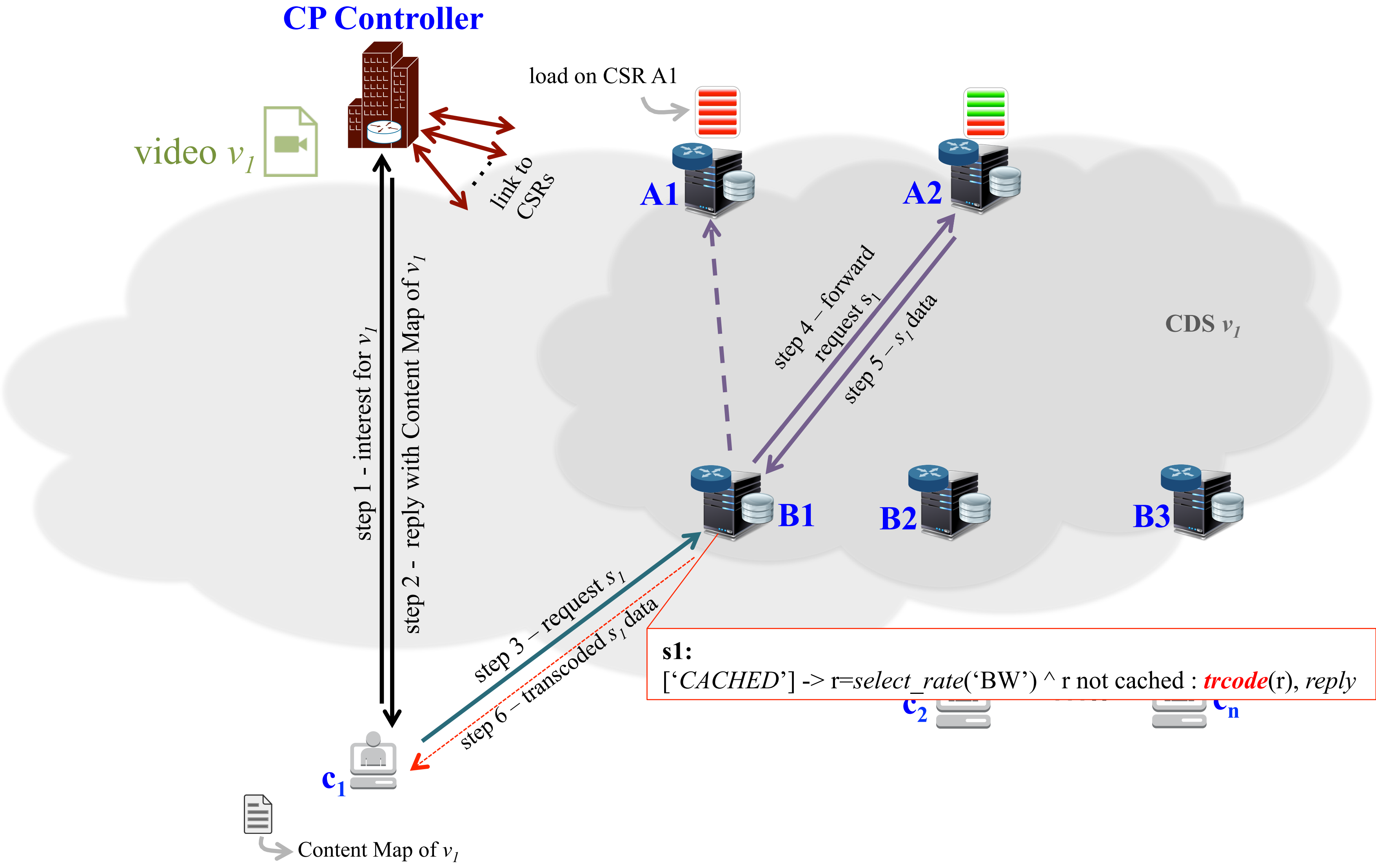
s1:
[‘*CACHED*’] -> r=*select_rate*(‘BW’) ^ r is cached : *reply*

Use Cases: Load-aware Forwarding



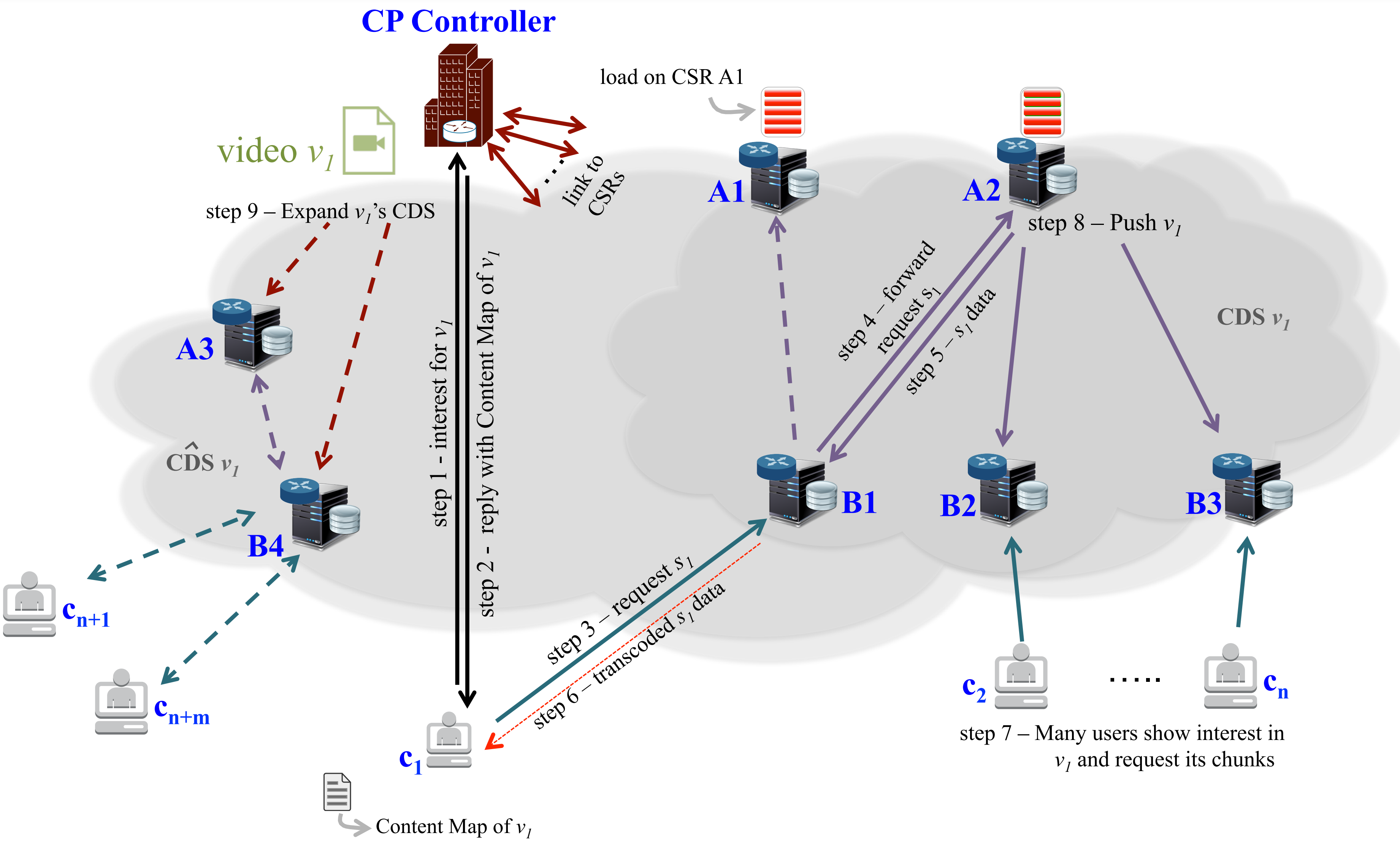
s1:
[‘*NOT_CACHED*’] -> *forward* (*select* ([A1, A2], **‘LOAD’**))

Use Cases: Dynamic Adaptation



s1:
[‘*CACHED*’] -> r=*select_rate*(‘BW’) ^ r not cached : ***trcode***(r), *reply*

Use Cases: Flash Crowds & Load Management


CP Controller
load on CSR A1
video v_1
link to CSRs
A1
A2
step 9 – Expand v_1's CDS
step 8 – Push v_1
A3
step 1 - interest for v_1
step 2 - reply with Content Map of v_1
step 4 – forward request s_1
step 5 – s_1 data
CDS v_1
$\hat{CDS}$ v_1
B1
B2
B3
B4
c_{n+1}
c_{n+m}
step 3 – request s_1
step 6 – transcoded s_1 data
c_1
c_2
.....
c_n
step 7 – Many users show interest in v_1 and request its chunks
Content Map of v_1

Conclusion

- CONIA: a *straw-man proposal* to argue for
  - The need for *namespace independence* for **complex** information delivery
  - The importance of providing *larger control* to content providers and considering the *network economics* for better content delivery
- Overview of CONIA's *content delivery* architecture
  - **Basic functions** of the key components
  - **Communications** between components
- Several use cases illustrated how CONIA allows content providers to *dynamically adapt* to user demands and *optimize content delivery* to meet user QoE expectations

References

1. Global Internet Phenomena Report - 2H2014
   Sandvine

2. Unreeling Netflix: Understanding and Improving Multi-CDN Movie Delivery
   Vijay K. Adhikari, Yang Guo, Fang Hao, Matteo Varvello, Volker Hilt, Moritz Steiner, and Zhi-Li Zhang
   In INFOCOM, 2012

3. Netflix OpenConnect
   https://openconnect.netflix.com/

Thank you

Questions?