A Totally Distributed Iterative Scheme for Web Service Addressing and Discovery

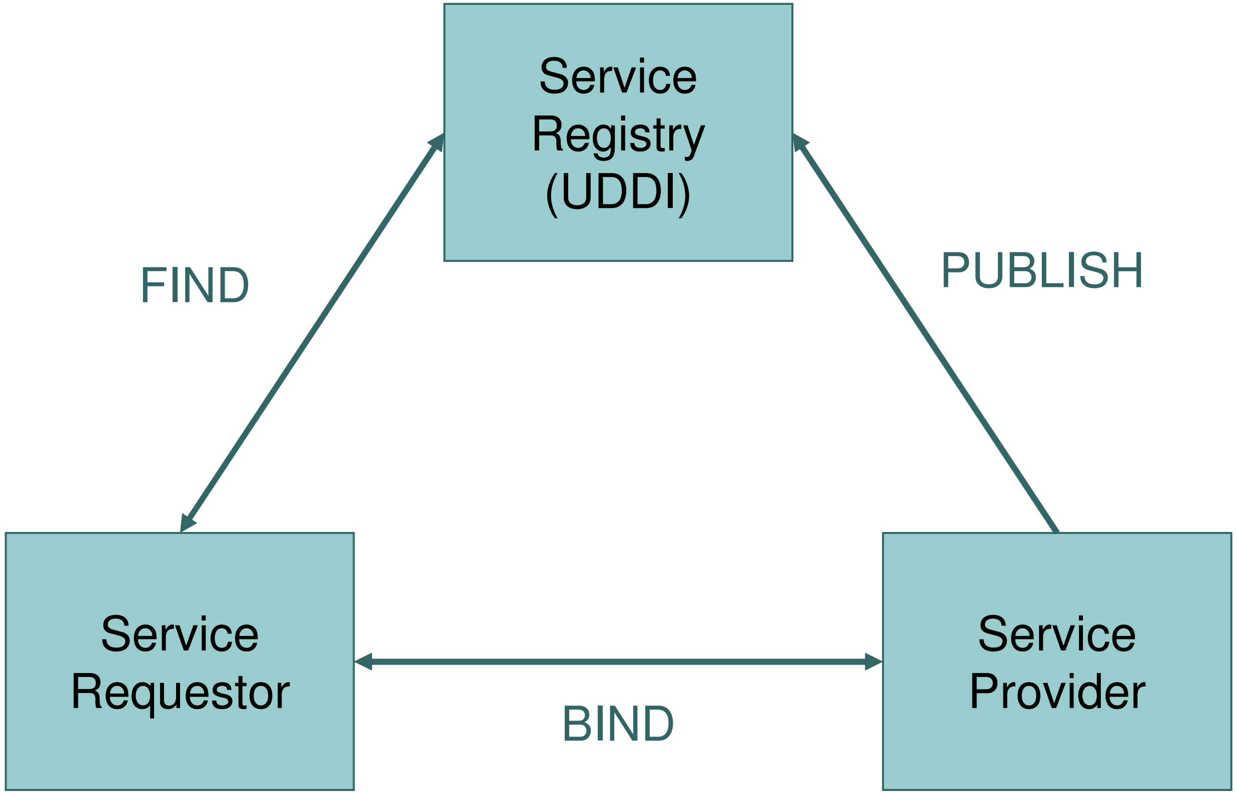
Firenze 11-12/06/2007

M.Milanesio, G.Ruffo, R.Schifanella
{milane, ruffo, schifane}@di.unito.it

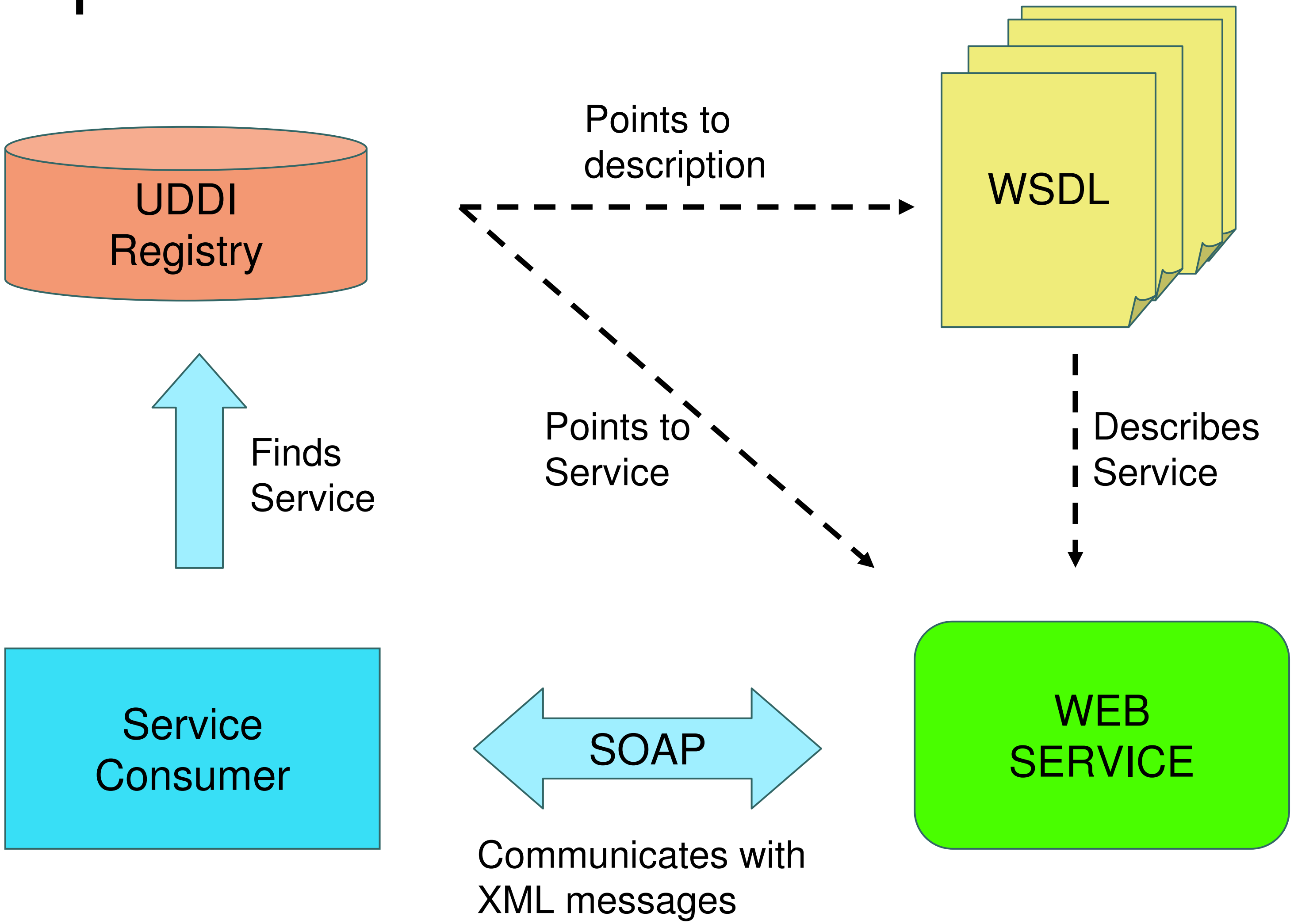
Problems

- Web Service Addressing & Discovery: UDDI decentralization
- Key-Based Routing on DHTs

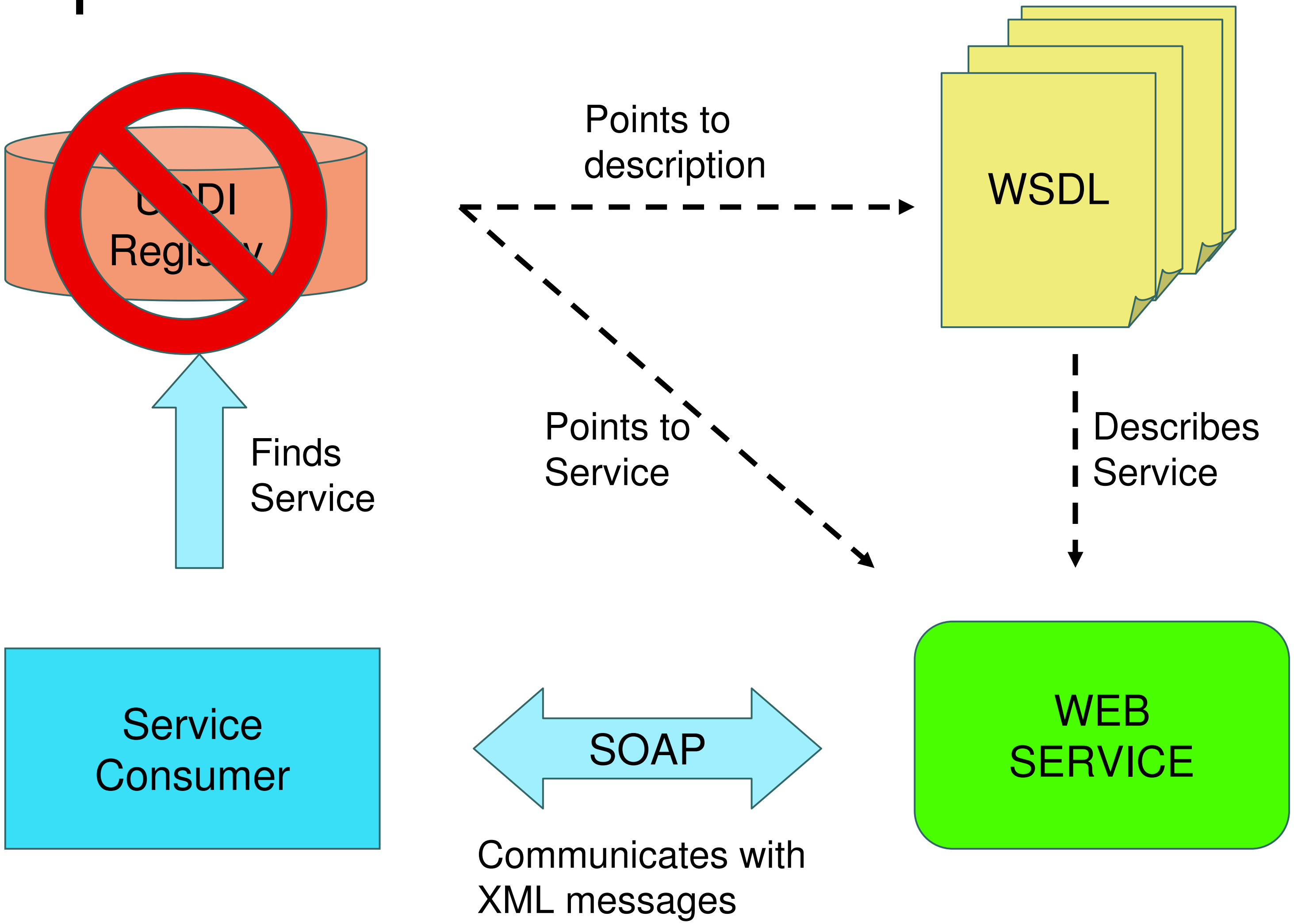
Web Services



Web Services



Web Services



UDDI Registry

Finds Service

Points to description

WSDL

Points to Service

Describes Service

Service Consumer

SOAP

Communicates with XML messages

WEB SERVICE

Structured P2P Systems

- DHT
- <key-value>
- Fixed Topology
- Replication
- Consistency
- Load Balance
- Fault-tolerance
- Robustness
- Scalability
- High churn rate

→ Chord-CAN-Kademlia-Pastry…

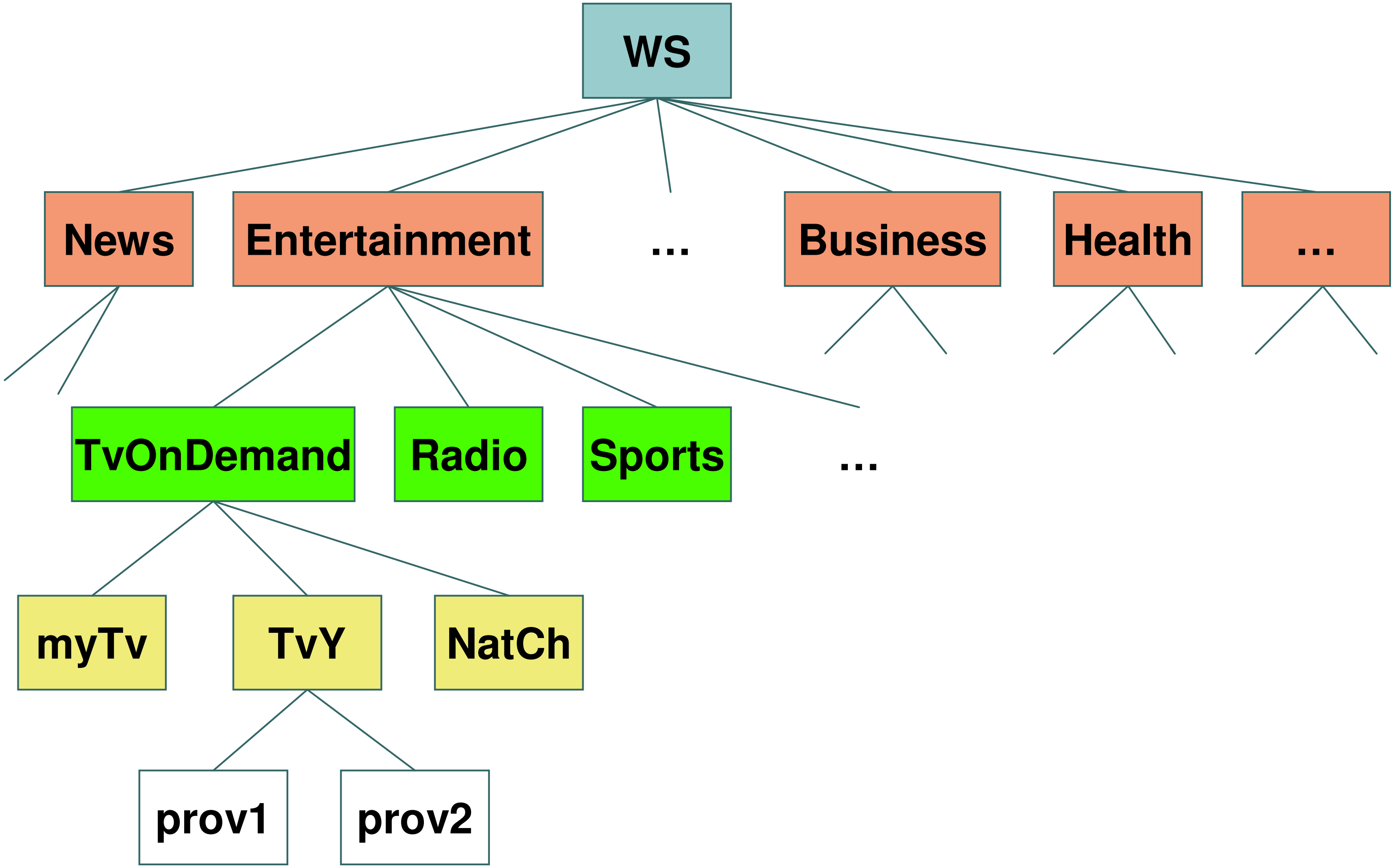
Key-Based Routing

- Hash values consistently calculated from a large **id space** (e.g., 128 bit).
- In order to find a resource (a node) user must know the identifier the resource is indexed with.
- This **cannot be assumable** at an application level.

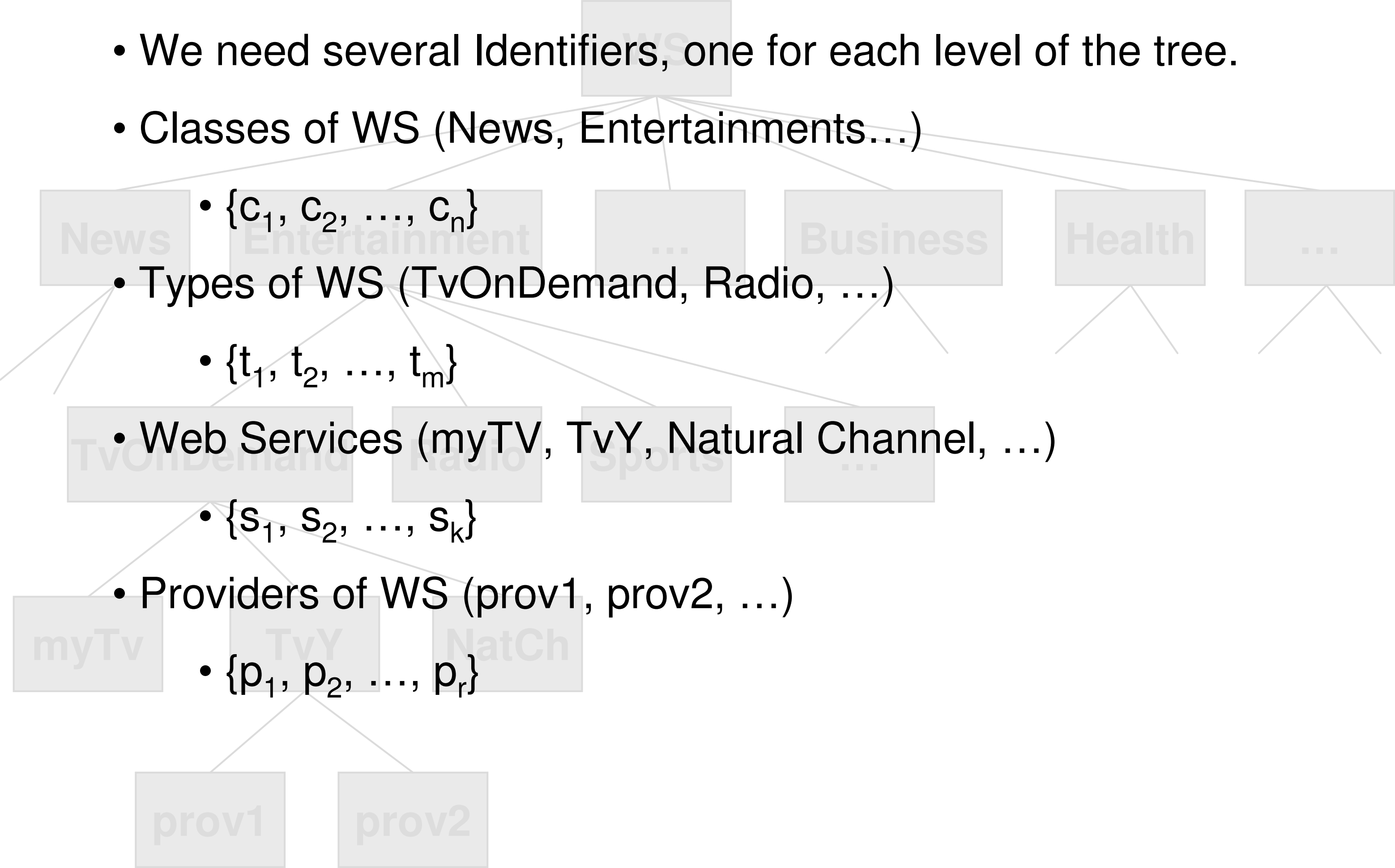
I-KoD

- **I**terative **K**ey-Based Routing **O**n **D**HTs
- Classification of Web Services in a *semantic* way.
- Class-Type-Name-Provider
- Tagging

Hierarchical Tagging

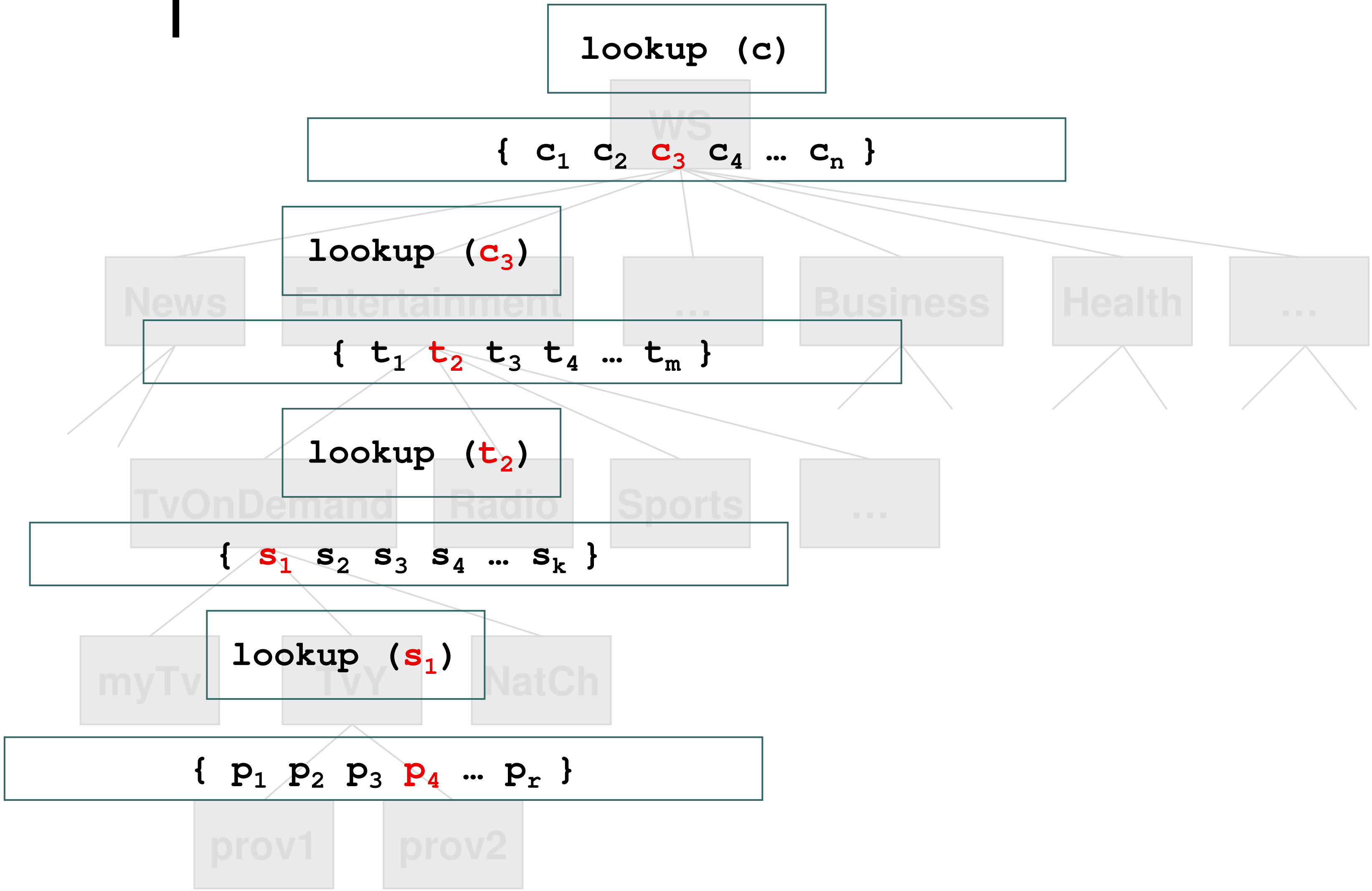


Navigation Tree



- We need several Identifiers, one for each level of the tree.
- Classes of WS (News, Entertainments…)
  - $\{c_1, c_2, \ldots, c_n\}$
- Types of WS (TvOnDemand, Radio, …)
  - $\{t_1, t_2, \ldots, t_m\}$
- Web Services (myTV, TvY, Natural Channel, …)
  - $\{s_1, s_2, \ldots, s_k\}$
- Providers of WS (prov1, prov2, …)
  - $\{p_1, p_2, \ldots, p_r\}$

Navigation Tree



lookup (c)

$\{\ c_1\ c_2\ c_3\ c_4\ \dots\ c_n\ \}$

lookup (c_3)

$\{\ t_1\ t_2\ t_3\ t_4\ \dots\ t_m\ \}$

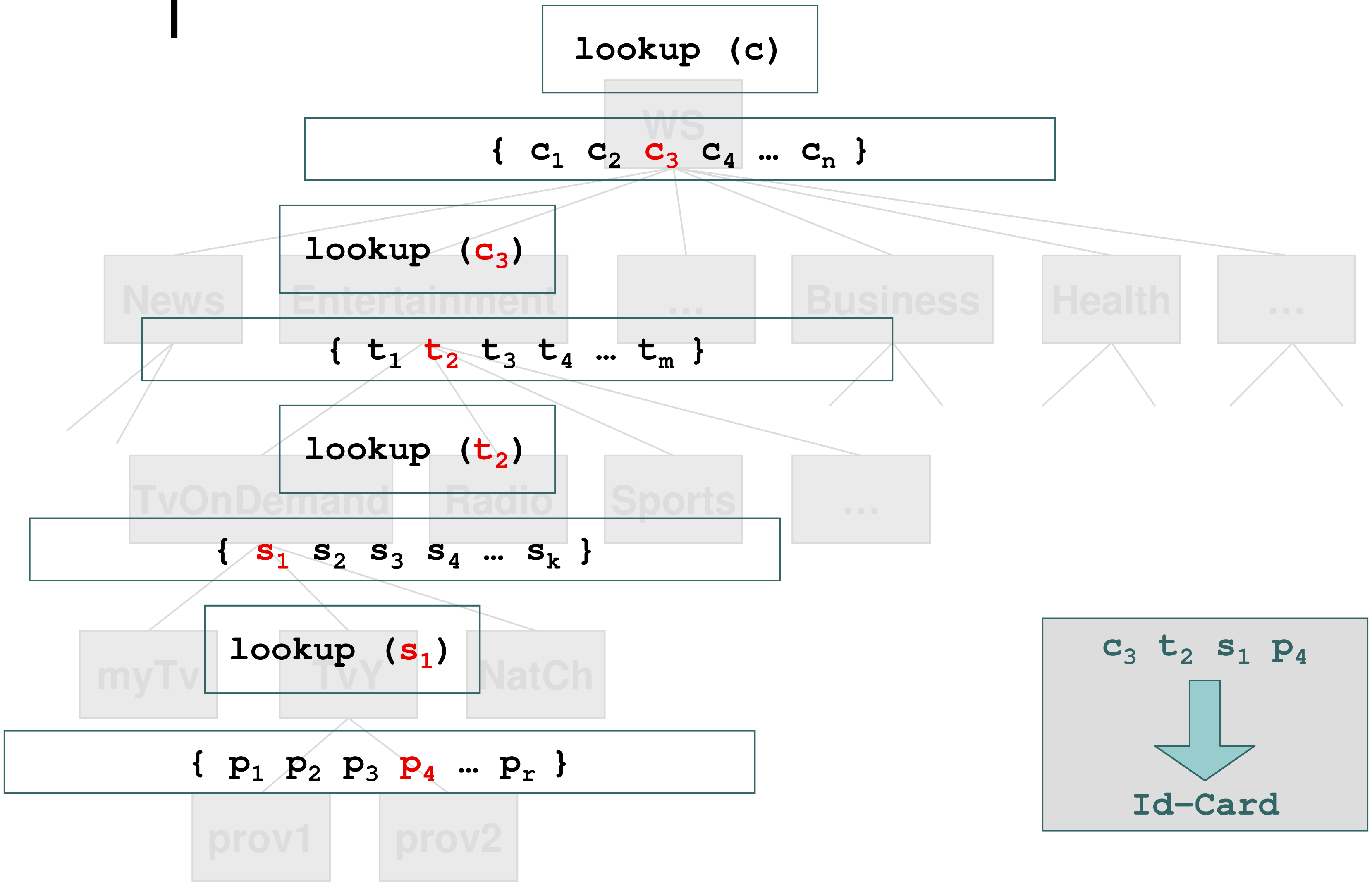
lookup (t_2)

$\{\ s_1\ s_2\ s_3\ s_4\ \dots\ s_k\ \}$

lookup (s_1)

$\{\ p_1\ p_2\ p_3\ p_4\ \dots\ p_r\ \}$

Navigation Tree



lookup (c)

{ c_1 c_2 c_3 c_4 … c_n }

lookup (c_3)

{ t_1 t_2 t_3 t_4 … t_m }

lookup (t_2)

{ s_1 s_2 s_3 s_4 … s_k }

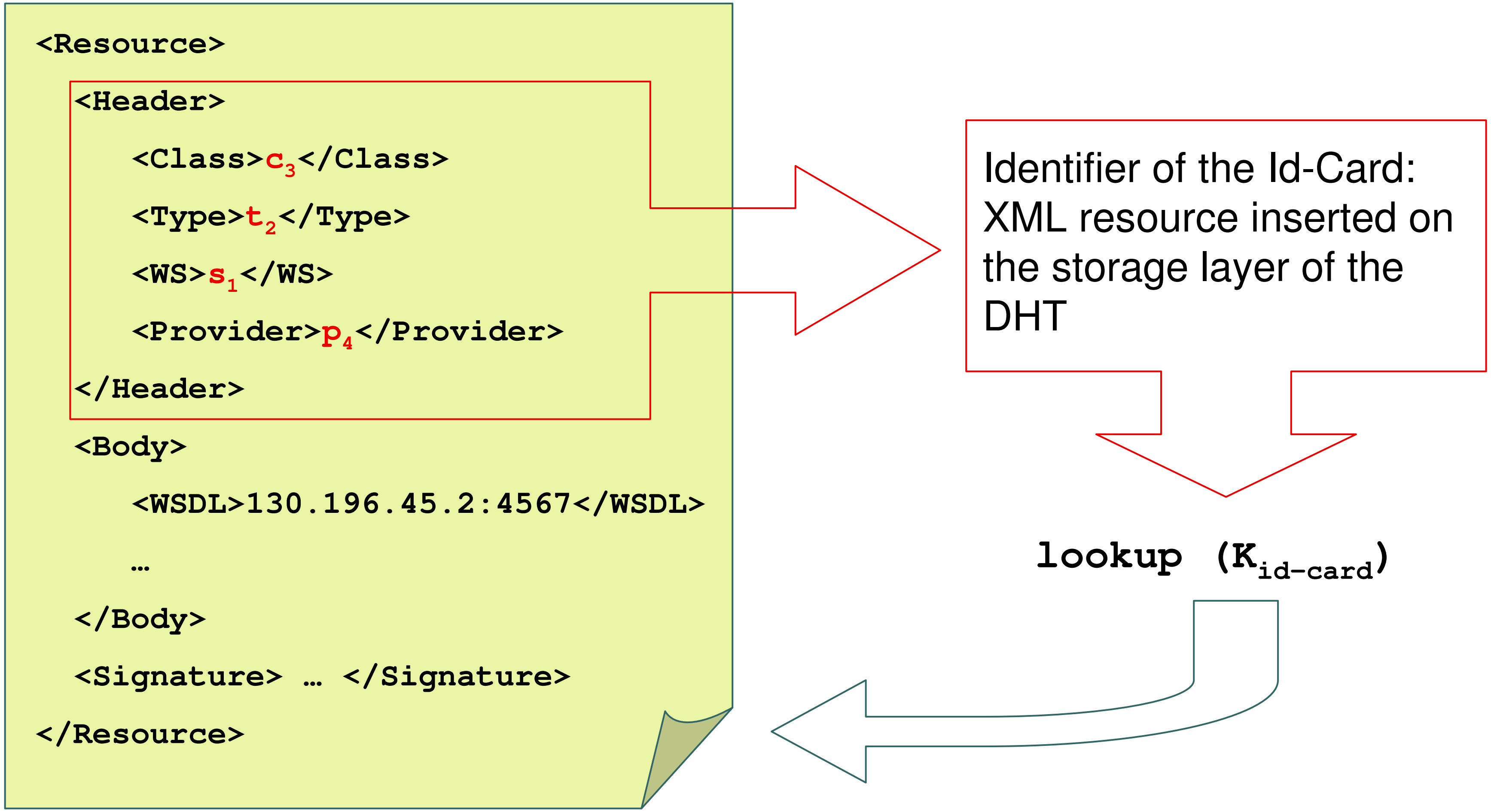
lookup (s_1)

{ p_1 p_2 p_3 p_4 … p_r }

c_3 t_2 s_1 p_4

Id-Card

Id-Card



<Resource>

<Header>

<Class>c_3</Class>

<Type>t_2</Type>

<WS>s_1</WS>

<Provider>p_4</Provider>

</Header>

<Body>

<WSDL>130.196.45.2:4567</WSDL>

…

</Body>

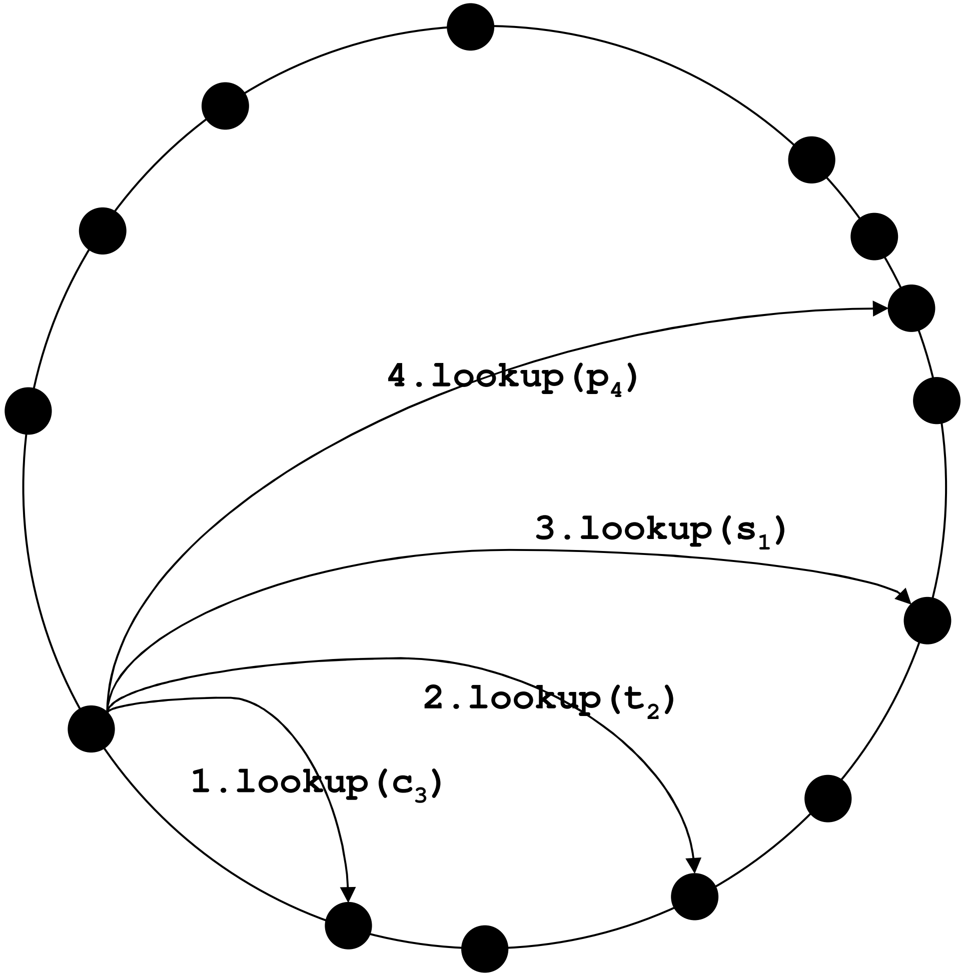
<Signature> … </Signature>

</Resource>

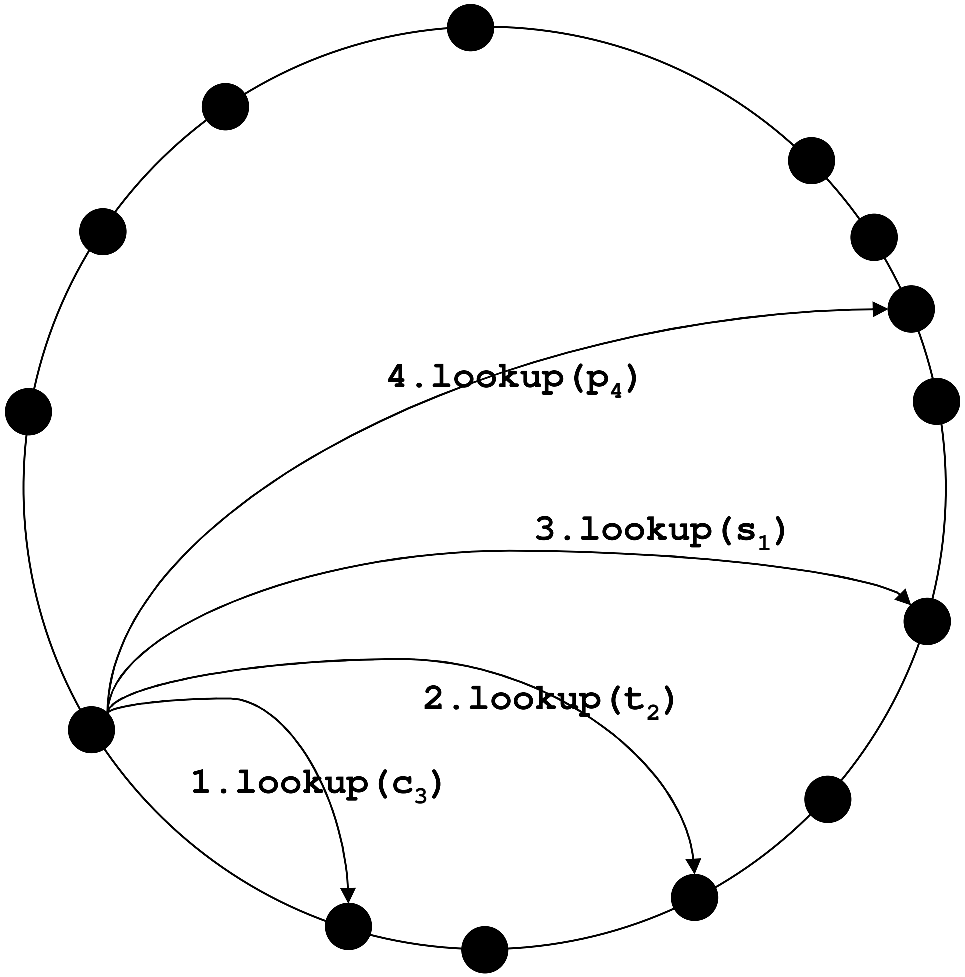
Identifier of the Id-Card: XML resource inserted on the storage layer of the DHT

lookup ($K_{id\text{-}card}$)

Iterative Key-Based Routing



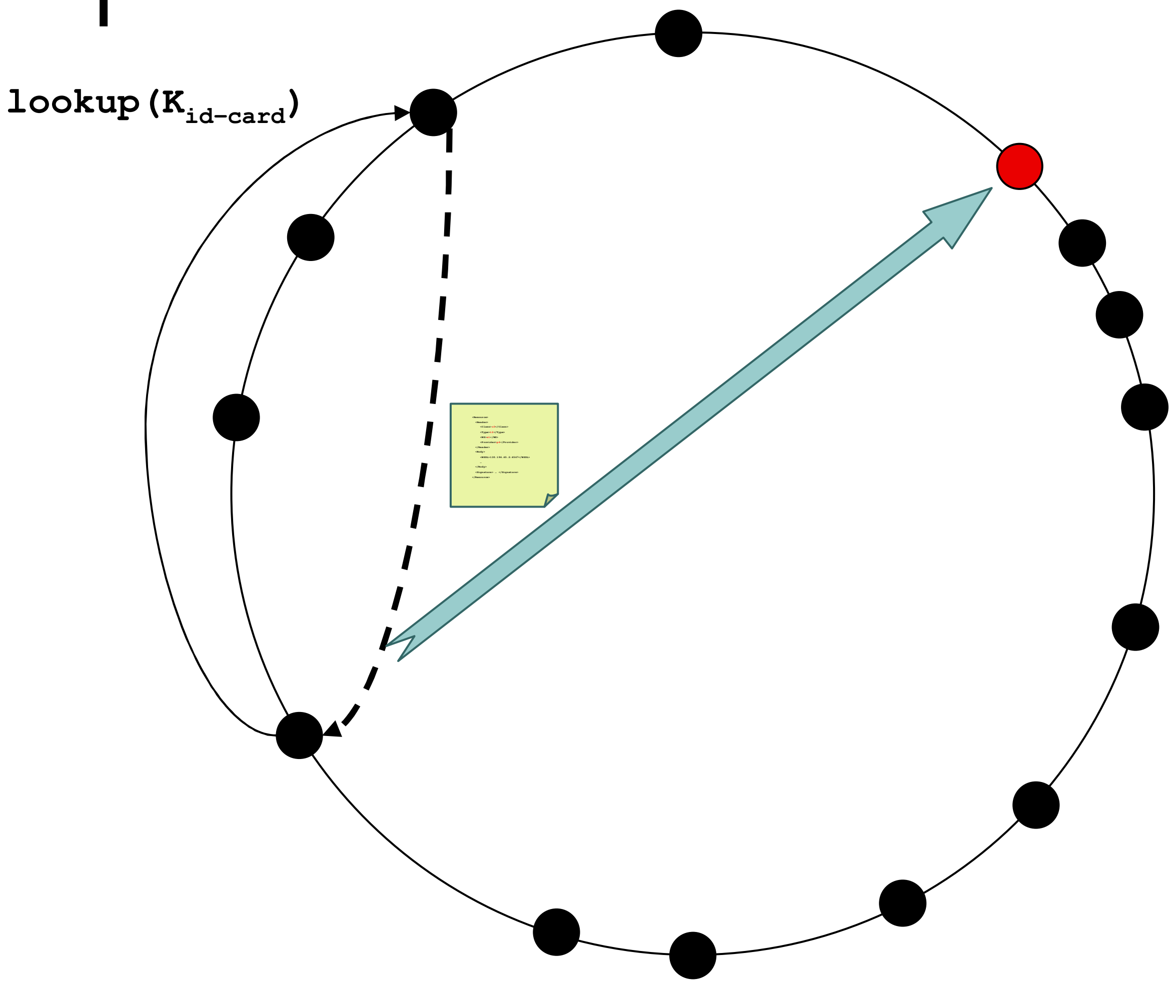
Iterative Key-Based Routing



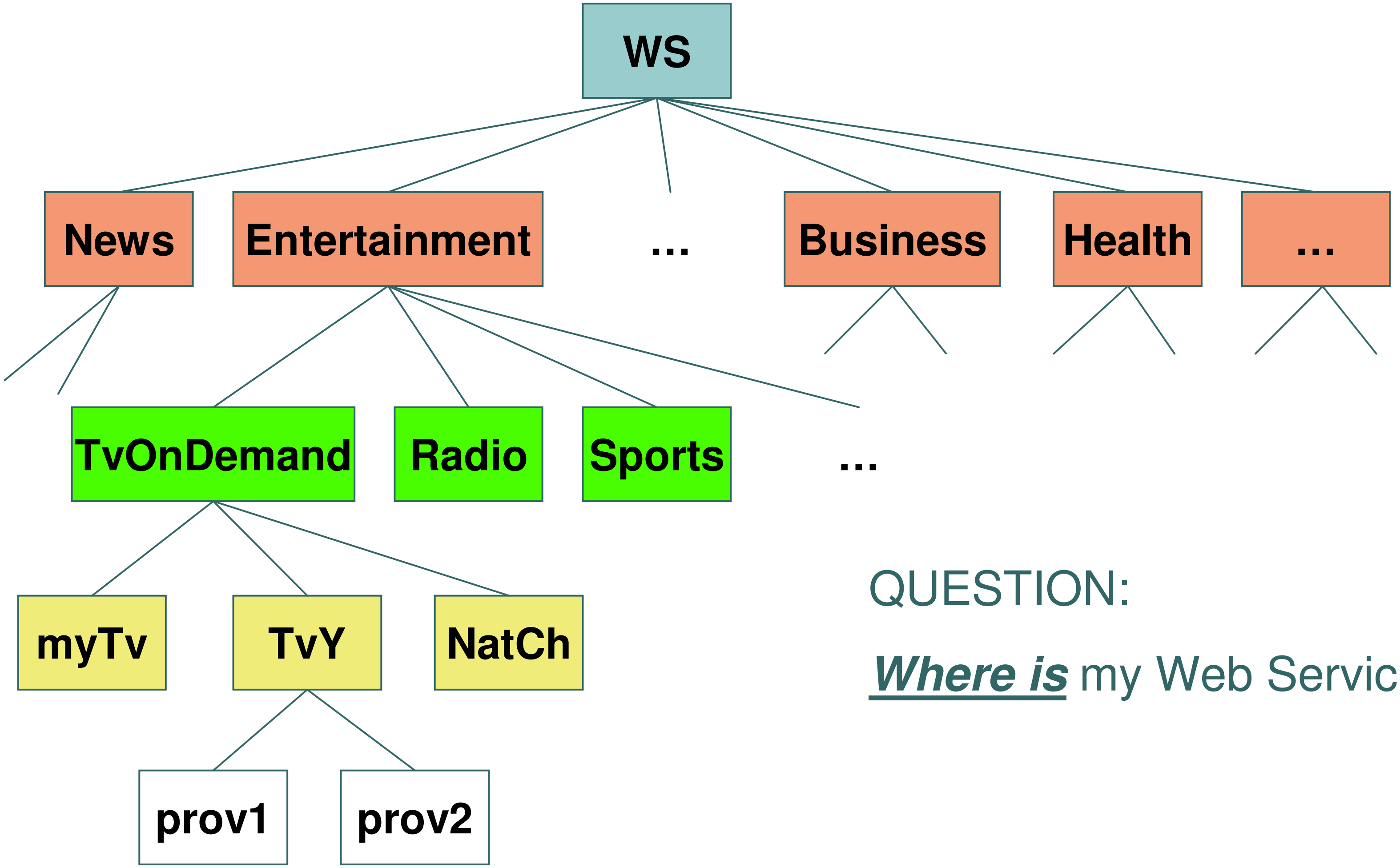
c_3, t_2, s_1, p_4

$K_{\text{id-card}}$

Iterative Key-Based Routing



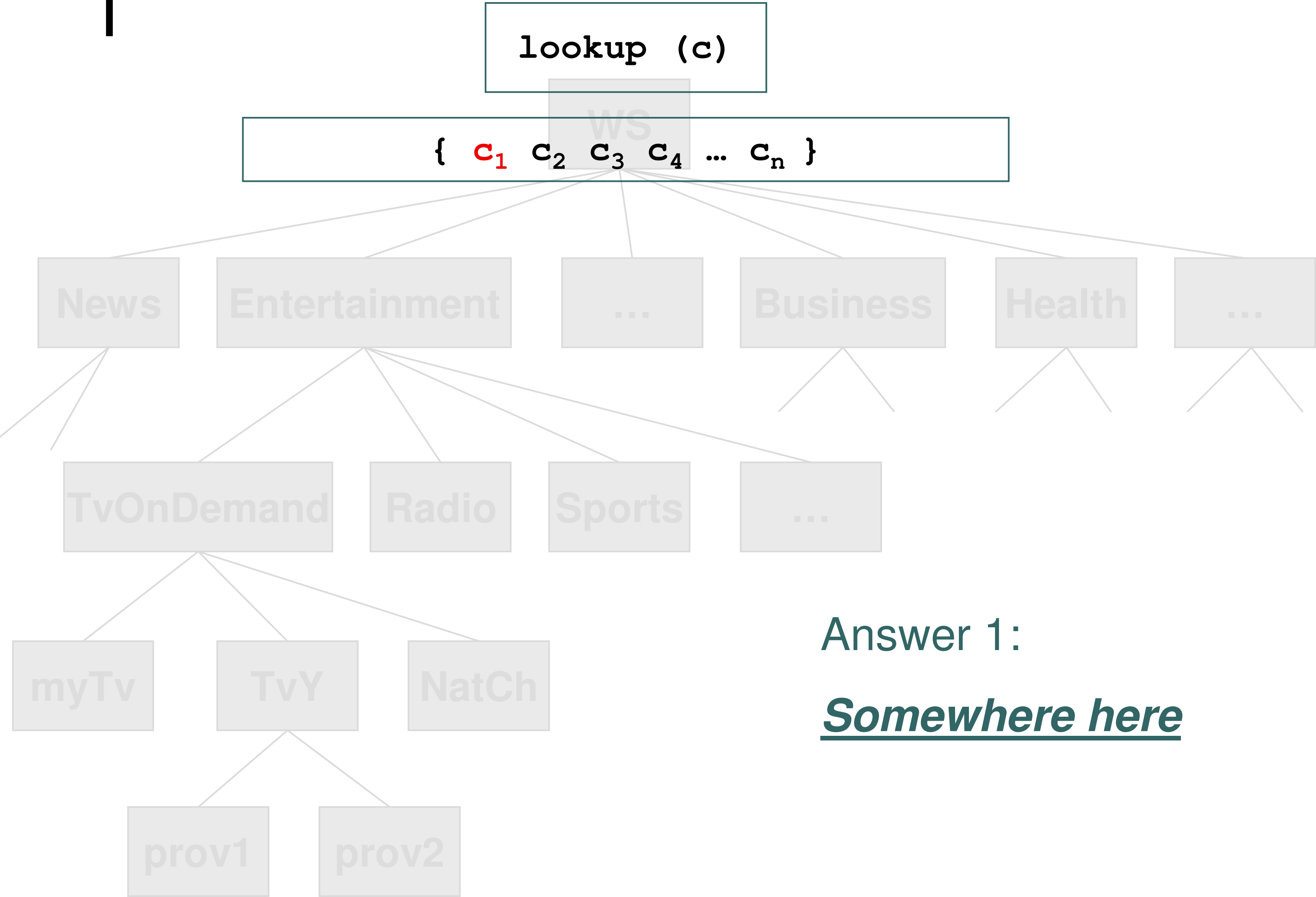
Insertion



QUESTION:

Where is my Web Service?

Insertion



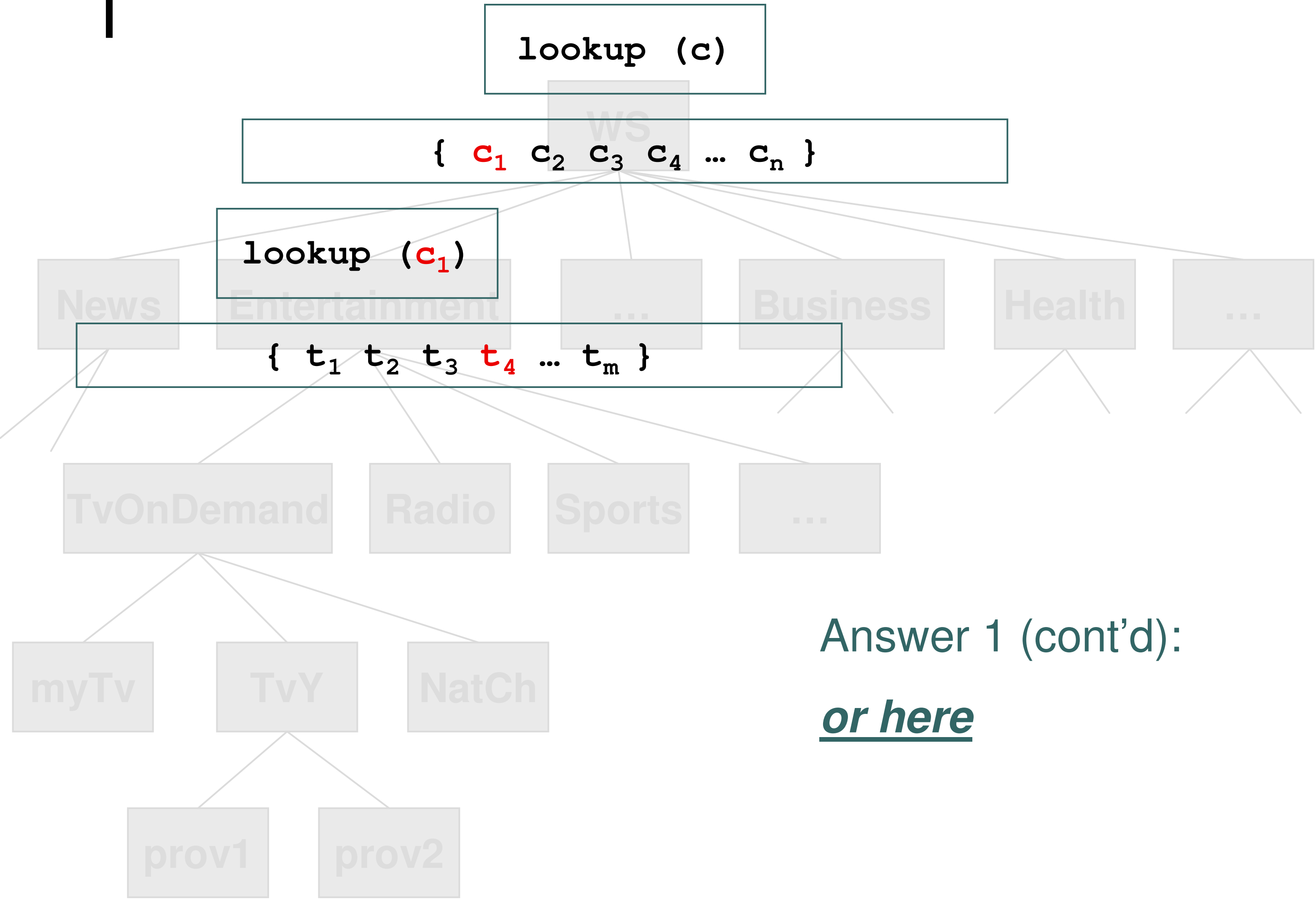
`lookup (c)`

$\{\ c_1\ \ c_2\ \ c_3\ \ c_4\ \dots\ c_n\ \}$

Answer 1:

Somewhere here

Insertion



lookup (c)

$\{ c_1\ c_2\ c_3\ c_4\ \dots\ c_n \}$

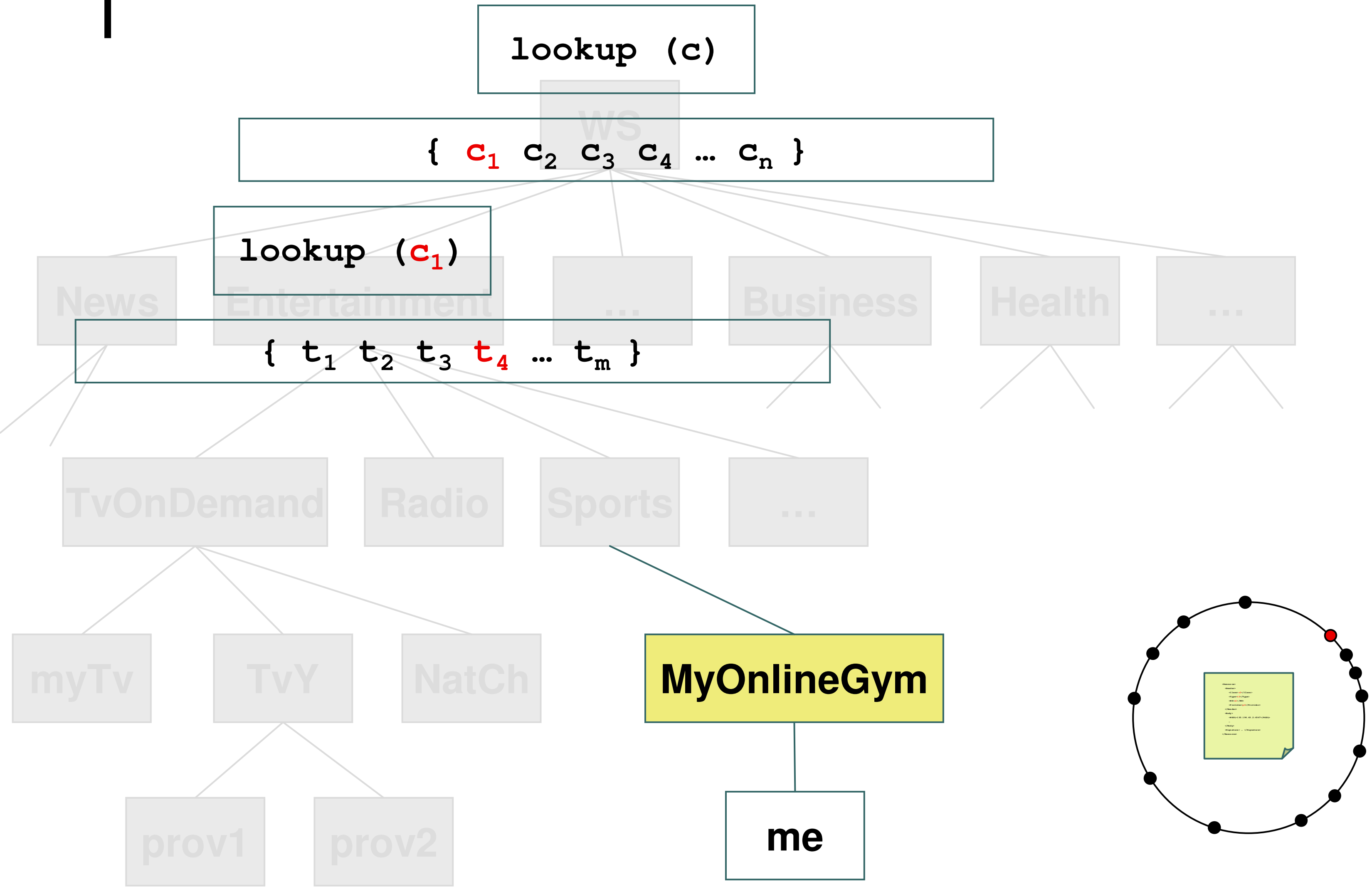
lookup (c_1)

$\{ t_1\ t_2\ t_3\ t_4\ \dots\ t_m \}$

Answer 1 (cont'd):

<u>or here</u>

Insertion



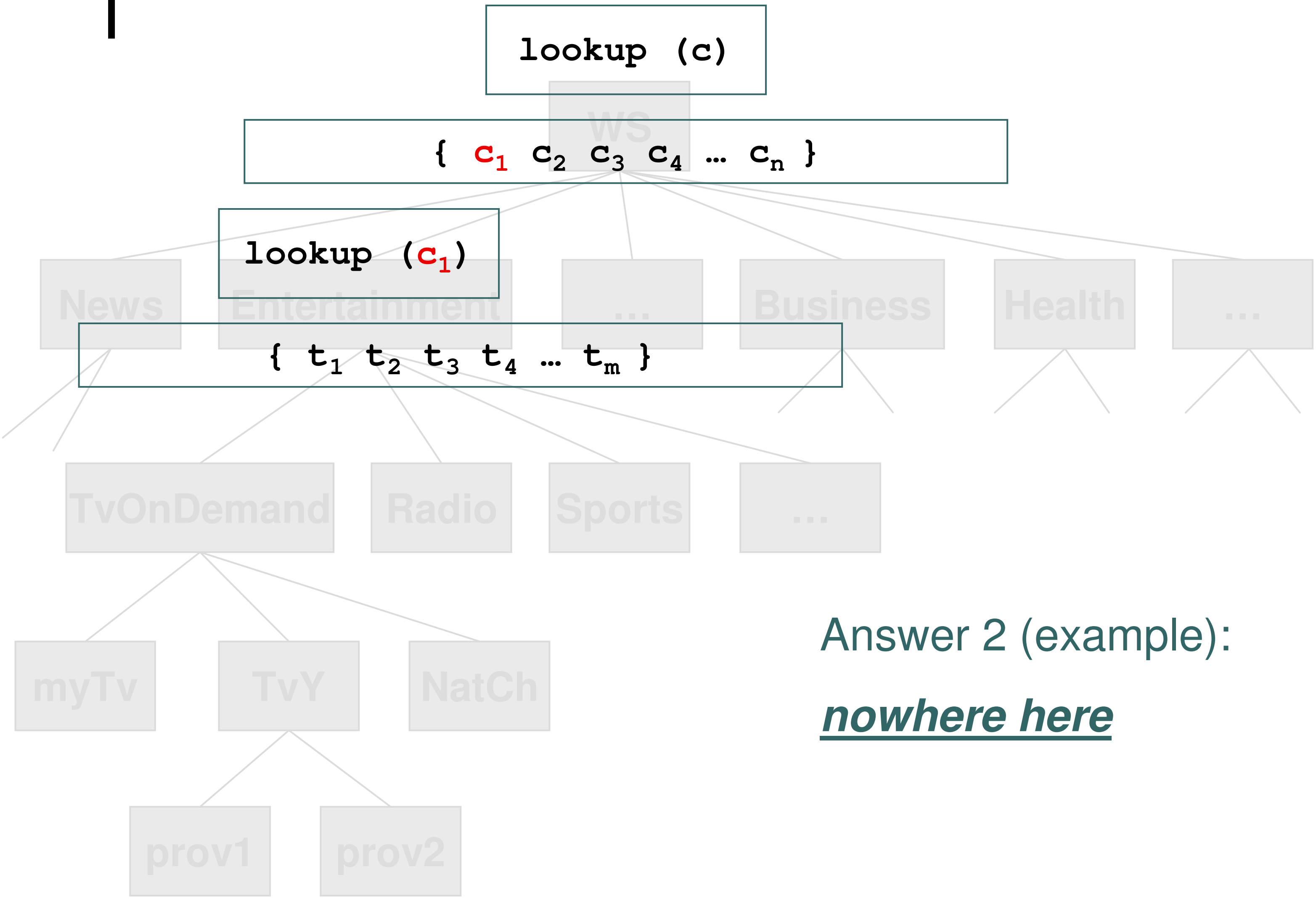
`lookup (c)`

$\{\ c_1\ c_2\ c_3\ c_4\ \dots\ c_n\ \}$

`lookup (c1)`

$\{\ t_1\ t_2\ t_3\ t_4\ \dots\ t_m\ \}$

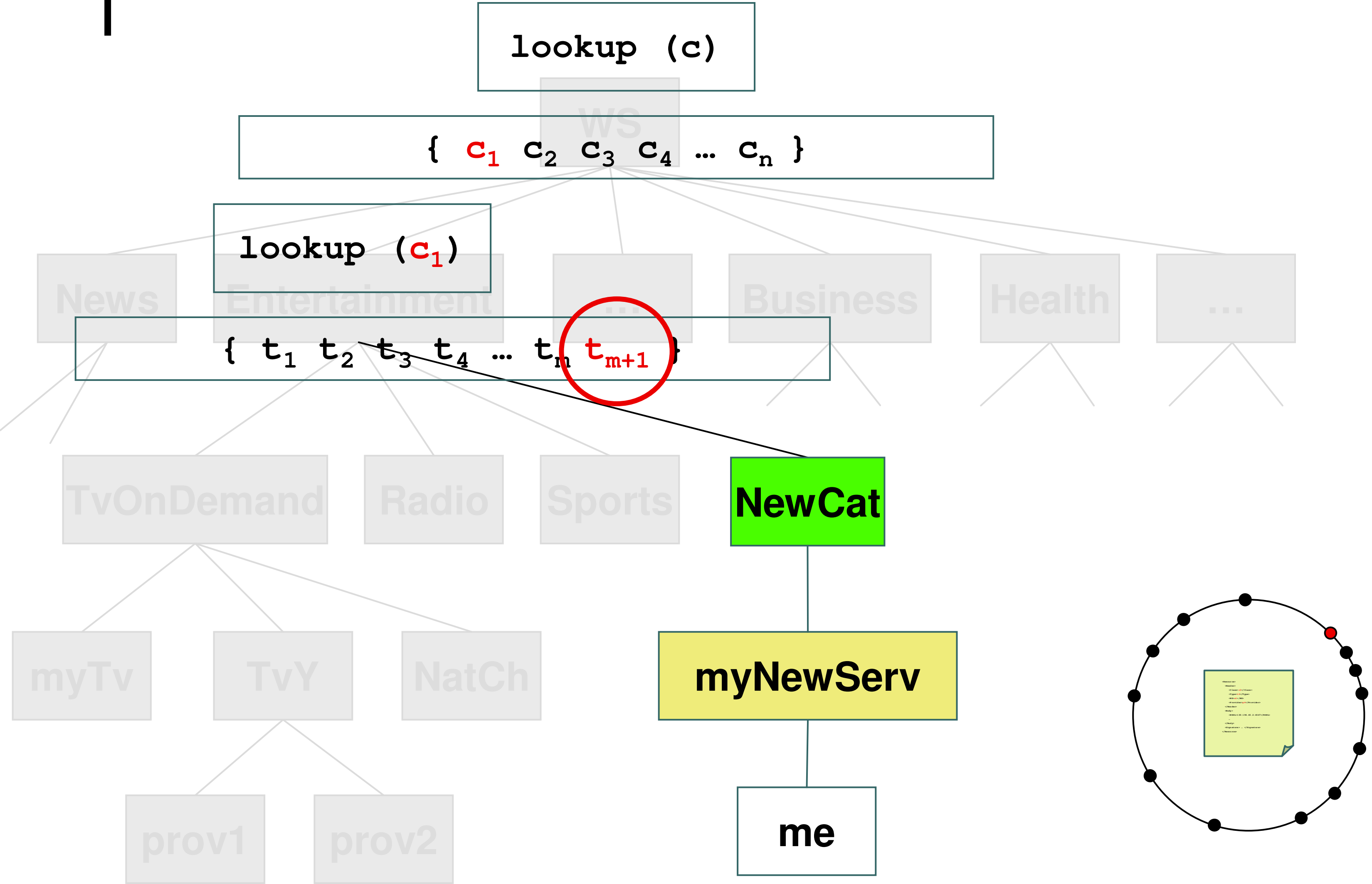
Insertion



Answer 2 (example):

nowhere here

Insertion



`lookup (c)`

`{` c_1 c_2 c_3 c_4 … c_n `}`

`lookup (`c_1`)`

`{` t_1 t_2 t_3 t_4 … t_n t_{m+1} `}`

NewCat

myNewServ

me

Features

- **key-to-key** mapping, instead of **key-to-value**.
- The **distinct paths** that can lead to a same resource will travel across **different indexes** (i.e., different keys).
- Service **update**, or new service insertion, will **not concern higher level brokers**, i.e., nodes responsible for more generic keys
- Users share their resources with their tags, generating an aggregated tag-index so-called **folksonomy**.
- Users can actively query the system, and the given classification can be **updated** and **extended** by service providers themselves, with the insertions of **new tags**.

To do

- **Simulative evaluation**
- **Prototype implementation**
- **Performance analysis**