A Tunneling Service Controller

Damien Saucez Benoit Donnet Olivier Bonaventure

Université Catholique de Louvain

September 11, 2007

Motivation

- The problem with the EID-to-RLOC mapping
- The quality of the EID-to-RLOC mapping is important
- The problem is more general

TSC Service

- TSC introduction
- TSC protocol
- TSC interactions
- TSC implementation

Conclusion

Motivation

- The problem with the EID-to-RLOC mapping
- The quality of the EID-to-RLOC mapping is important
- The problem is more general

TSC Service

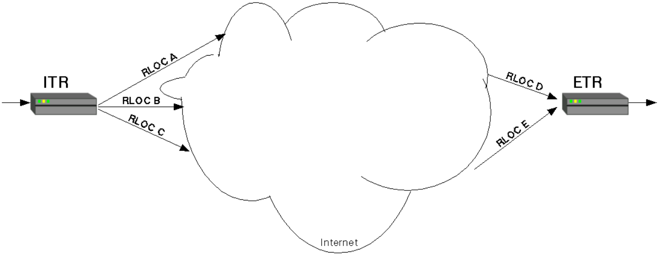
- TSC introduction
- TSC protocol
- TSC interactions
- TSC implementation

Conclusion

The facts : Both **ITR** and **ETR** routers can have more than one **RLOC** associated to them.

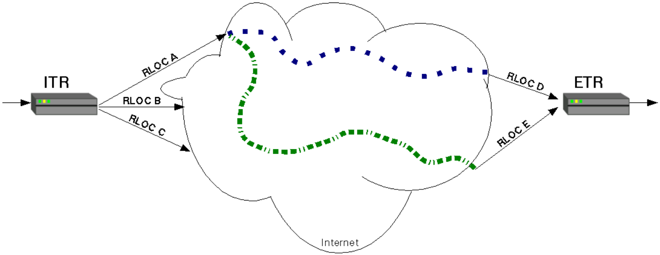
The facts : Both **ITR** and **ETR** routers can have more than one **RLOC** associated to them.

The problem : The path followed by the messages from an ITR to an ETR depends on the associated RLOC for the tunnel.



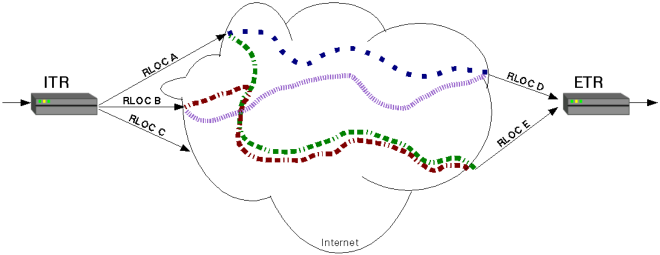
The facts : Both **ITR** and **ETR** routers can have more than one **RLOC** associated to them.

The problem : The path followed by the messages from an ITR to an ETR depends on the associated RLOC for the tunnel.



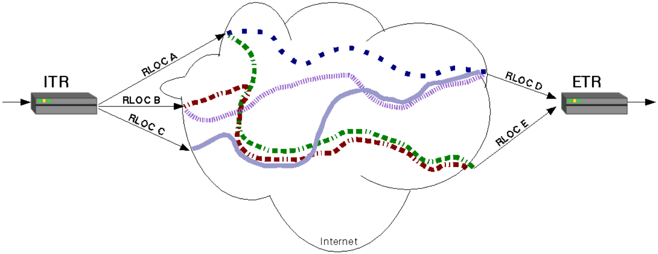
The facts : Both **ITR** and **ETR** routers can have more than one **RLOC** associated to them.

The problem : The path followed by the messages from an ITR to an ETR depends on the associated RLOC for the tunnel.



The facts : Both **ITR** and **ETR** routers can have more than one **RLOC** associated to them.

The problem : The path followed by the messages from an ITR to an ETR depends on the associated RLOC for the tunnel.



The quality of the EID-to-RLOC mapping is important

- The source and destination RLOCs partially define the path of a LISP tunnel.
- The construction of the EID-to-RLOC Database must take this information into account, but:
  - for scalability issues, the LISP routers cannot know every path (quadratic in the number of RLOCs),
  - for performances issues, the LISP routers cannot analyse paths on demand.

⇒ An independent service that can identify the best paths based on the source and destination RLOCs would be a solution.

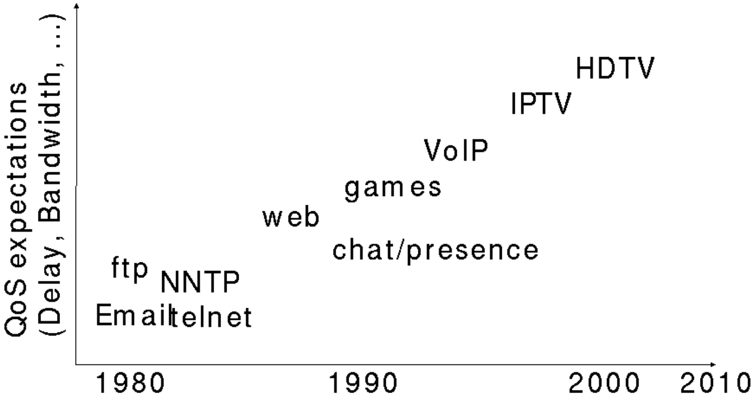
The problem is more general

Multihoming : The choice of the ISP can have impacts on performances and costs.

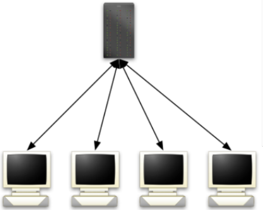
Overlays : Modern overlays (CAN, P2P. . .) use their own routing tables and take control over the routing.

Mirrored content : Cost functions of the overlays are seldom based on the underlay.

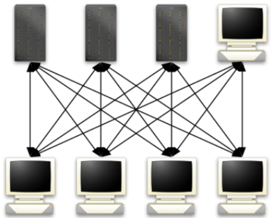
Applications' expectations are evolving



Servers are evolving



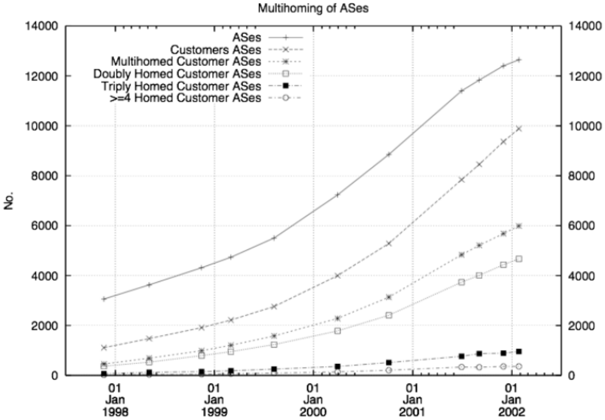
Past: One Content = One Server



Today: Many servers offer the same content

Increasing interest for multihoming

At least 60% of the stubdomains are multihomed ([Agarwal03]).



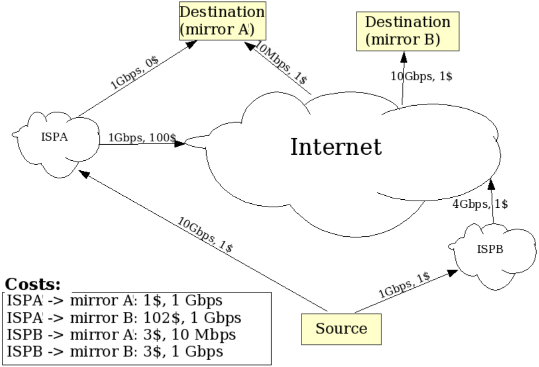
ISPs' networks could become hard to manage

The behaviour of modern applications can destabilize ISPs:

- The objectives of the underlays and the applications are in conflicts (low cost and stability vs low cost, flexibility and performances).
- Transmissions on the underlay are not optimized (one message can cross the same link more than once).
- It may become hard to apply TE policies.
- Measurements are frequently redundant because applications do not exchange information together (ping storms, bandwidth probes, traceroutes . . .).
- . . .

An example of conflict, an observation

Different choices with different costs and different performances

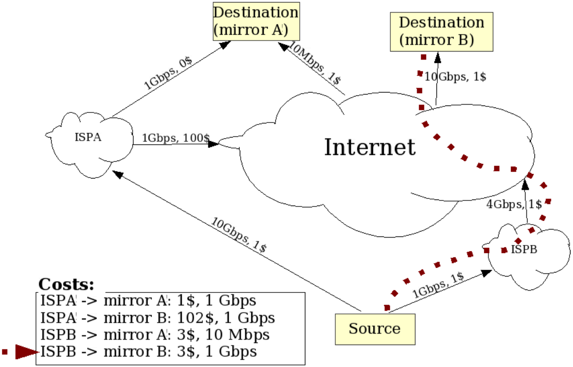


Costs:

ISPA -> mirror A: 1$, 1 Gbps
ISPA -> mirror B: 102$, 1 Gbps
ISPB -> mirror A: 3$, 10 Mbps
ISPB -> mirror B: 3$, 1 Gbps

An example of conflict, an observation

The application would prefer mirror B through ISP B. . .



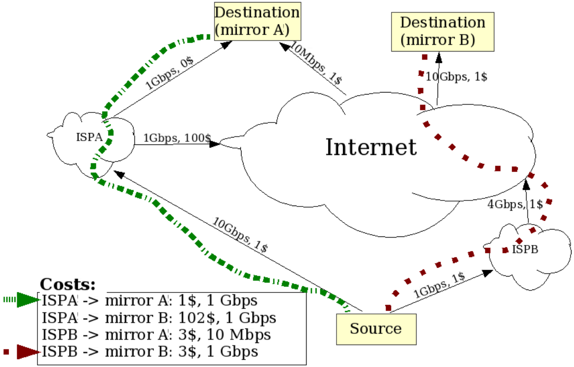
Costs:

ISPA -> mirror A:	1$, 1 Gbps
ISPA -> mirror B:	102$, 1 Gbps
ISPB -> mirror A:	3$, 10 Mbps
ISPB -> mirror B:	3$, 1 Gbps

An example of conflict, an observation

The application would prefer mirror B through ISP B. . .

. . . but the underlay would prefer the mirror A trough ISP A.



Costs:

ISPA -> mirror A: 1$, 1 Gbps

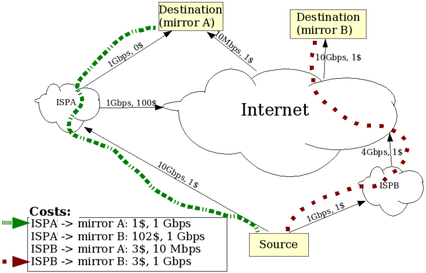
ISPA -> mirror B: 102$, 1 Gbps

ISPB -> mirror A: 3$, 10 Mbps

ISPB -> mirror B: 3$, 1 Gbps

An example of conflict, identify the problem

- The application uses its own cost function that maximize the bandwidth and makes a random choice for tie-break.
- The underlay prefers ISP A for cost reasons and the ISP A prefers mirror A as it is inside the network.



Costs:
ISPA -> mirror A: 1$, 1 Gbps
ISPA -> mirror B: 102$, 1 Gbps
ISPB -> mirror A: 3$, 10 Mbps
ISPB -> mirror B: 3$, 1 Gbps

A good choice of the path is important

The problem comes from the **lack of communication** between the overlay (application) and the underlay (ISP).

- The applications cannot obtain all the knowledge of the underlay (scalability and security issues).
- The underlay cannot support all the functionality of the applications (performances, security and stability issues).

⇒ An independent service that can identify the best paths based on the source and destination addresses would be a solution.

The challenge

- Always-on, scalable service,
- Provide an API that can be queried by applications to obtain a good prediction of the best path to follow,
- Efficiently perform measurements,
- *Ensure security and reliability*.

Motivation

The problem with the EID-to-RLOC mapping

The quality of the EID-to-RLOC mapping is important

The problem is more general

TSC Service

TSC introduction

TSC protocol

TSC interactions

TSC implementation

Conclusion

TCS Service

Propose a paths selection service that can be queried by the underlay and the overlay:

1. The **client** gives a list of source addresses, a list of destination address and a quality of service.
2. The **server** returns an ordered list of couples of source/destination addresses.
   - The first entry in the ordered the more profitable choice accordingly to the selected QoS.
   - The list may not contain all the possible couples.
3. Servers are **stateless** (remember no information about clients).
4. Servers can work in **anycast**[1].

[1] The protocol to synchronise servers is out of the scope of this presentation

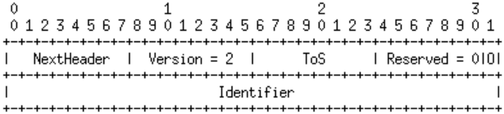
TSC protocol: overview

TSC messages are a collection of extensions:

- 5 different extensions:
  - TSC_HEADER (both directions),
  - TSC_REQUEST (client to server),
  - TSC_RESPONSE (server to client),
  - TSC_ERROR (server to client),
  - TSC_IN_TSC (both directions).
- Every TSC message must begin with a TSC_HEADER.

TSC protocol: `TSC_HEADER`

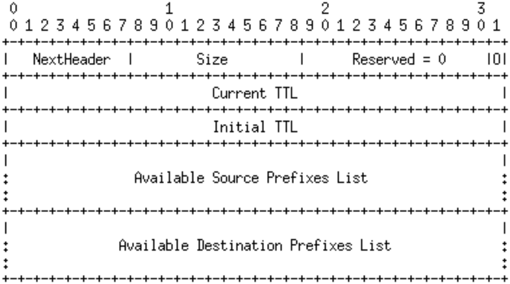
Specifies the structure of the TSC message



```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   NextHeader  |  Version = 2  |      ToS      | Reserved = 0|0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Identifier                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

TSC protocol: TSC_REQUEST

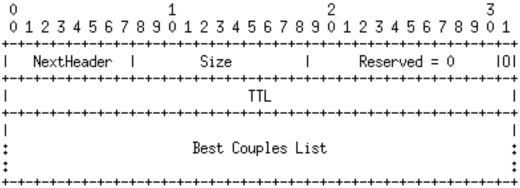
Asks for the best choices among the possible choices



```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   NextHeader  |       Size           |     Reserved = 0      |0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Current TTL                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Initial TTL                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                Available Source Prefixes List                 :
:                                                               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:              Available Destination Prefixes List              :
:                                                               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

TSC protocol: TSC_RESPONSE

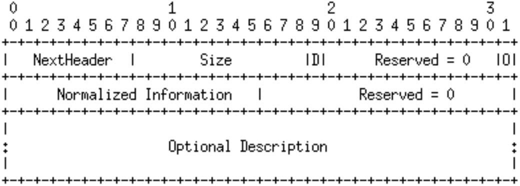
Gives a list of the best choices for a given TSC_REQUEST



```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   NextHeader  |       Size          |     Reserved = 0      |O|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             TTL                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                       Best Couples List                       :
:                                                               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

TSC protocol: TSC_ERROR

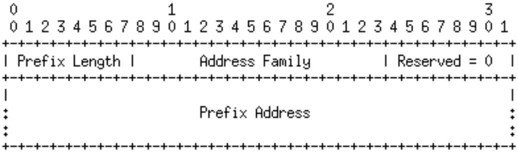
Specifies the nature of an error



```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   NextHeader  |        Size         |D|      Reserved = 0   |0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Normalized Information   |           Reserved = 0        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                   Optional Description                        :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

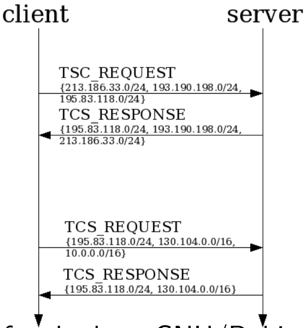
TSC protocol: prefixes

TSC works with prefixes as they are a generalization of address (a zero-length prefix is an address)



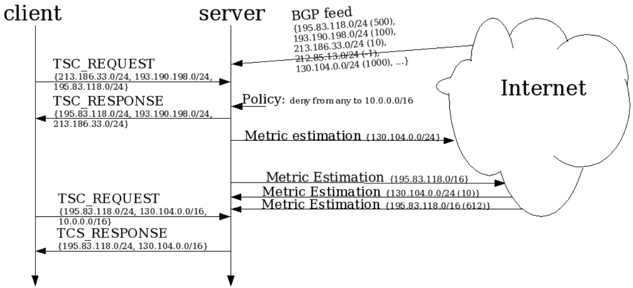
```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Prefix Length |         Address Family        | Reserved = 0  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                        Prefix Address                         :
:                                                               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

TSC as a blackbox



1. The client asks for the best GNU/Debian FTP mirror, Belnet is the more interesting one.
2. The client asks for a particular service, `10.0.0.0/24` does not appear in the response.

The core of TSC: Metrics



- As a first approximation, use immediately available estimators (passive metrics),
- For frequently used prefix, perform active measurements to refine the decision.

⇒ The client must never wait more than RTT to server + lookup time in server.

The core of TSC: Passive Metrics

Impossible to actively measure the quality of every path ⇒ use local information:

- BGP information like **localpref** reflects the ISP policies,
- IGP costs gives estimation about the interest of paths,
- Firewall configuration can limit the connectivity of the network,
- Administrative policies and SLAs gives information about paths preferences,
- Network Coordinates Systems gives a rough approximation of paths latency,
- ...

⇒ immediately available information gives enough information to order paths.

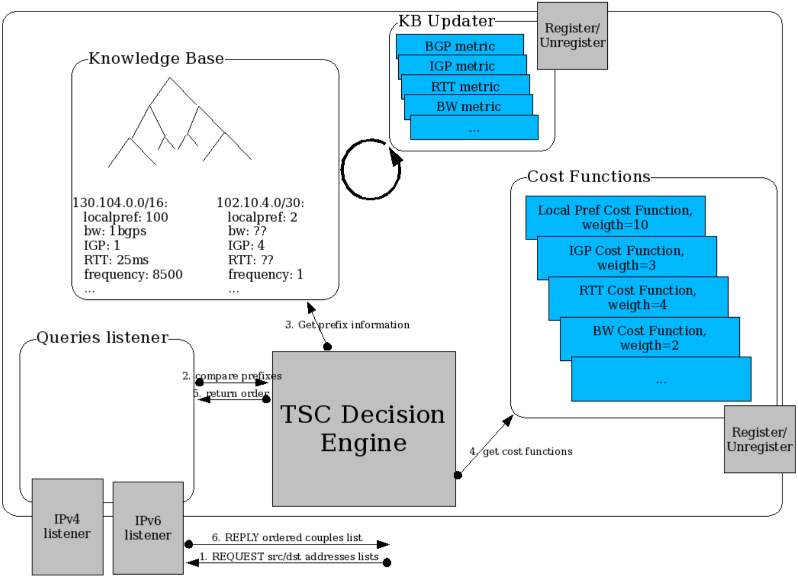
The core of TSC: Active Metrics

Few prefixes represent most of the traffic, mutuality of measurements for such prefixes can globally improve the quality of the network at low cost.

- Pings give information about the latency of paths;
- Traceroutes inform about the quality of paths, number of routers hops, etc;
- Many applications use a huge amount of bandwidth, bandwidth probes could be interesting;
- RT applications needs stable links (e.g., low jitter), measuring such information can improve the quality of RT traffic;
- . . .

TSC in LISP

- TSC servers implement EID-to-RLOC databases,
- ITR and ETR are TSC clients and update their cache with TSC_REQUESTs and TSC_RESPONSEs.
- The EID-to-RLOC Cache is updated only when the TTL of the TSC_RESPONSE is expired.



KB Updater

BGP metric

IGP metric

RTT metric

BW metric

...

Register/
Unregister

Knowledge Base

130.104.0.0/16:
localpref: 100
bw: 1bgps
IGP: 1
RTT: 25ms
frequency: 8500
...

102.10.4.0/30:
localpref: 2
bw: ??
IGP: 4
RTT: ??
frequency: 1
...

Cost Functions

Local Pref Cost Function,
weigth=10

IGP Cost Function,
weigth=3

RTT Cost Function,
weigth=4

BW Cost Function,
weigth=2

...

Register/
Unregister

Queries listener

3. Get prefix information

2. compare prefixes

5. return order

TSC Decision Engine

4. get cost functions

IPv4
listener

IPv6
listener

6. REPLY ordered couples list

1. REQUEST src/dst addresses lists

Motivation

- The problem with the EID-to-RLOC mapping
- The quality of the EID-to-RLOC mapping is important
- The problem is more general

TSC Service

- TSC introduction
- TSC protocol
- TSC interactions
- TSC implementation

Conclusion

- In the modern Internet, many paths are possible for a given content (redundancy of information, multihoming. . .),
- The choices of the paths are not always optimal as decision are made without enough information,
- TSC unifies the decision between the underlay and the path selection applications: The underlay can optimize the resources consumption.
- TSC receives a list of source addresses and destination addresses and returns a list of best couples.
- TSC can be used to improve performances of both underlay and overlay.
- TSC can be use to make ”proactive” load-balancing.

Our questions:

- Some idea for active and passive metrics?
- How to improve the protocol?
- How to improve the ordering of paths?
- Interactions with applications?
- What is missing?