Universidad
Politécnica
de Cartagena

git
Ingeniería Telemática

Net2Plan: The open-source network planner

José Luis Izquierdo Zaragoza
josel.izquierdo@upct.es
http://ait.upct.es/~jlizquierdo

Politecnico di Milano

Milano, Italy, November 22-27, 2013

This research was partially supported by a predoctoral FPU fellowship from the Spanish Ministry of Education, Culture and Sport (reference no. FPU12/04571) and a mobility grant from Universidad Politécnica de Cartagena (PMPDI-UPCT-2013)

Outline

Universidad Politécnica de Cartagena

1. Introduction
   - About us
   - Motivation
2. Net2Plan
   - Features
   - Tools
   - Website
3. Modeling network designs
4. What's next?

Outline

1. Introduction
   - About us
   - Motivation
2. Net2Plan
   - Features
   - Tools
   - Website
3. Modeling network designs
4. What's next?

About us I

Introduction

- Universidad Politécnica de Cartagena, UPCT (1998)

About us II

Introduction

- Telematics Engineering Group, GIT (1999)
  - Group leader: Prof. Joan García Haro
  - 20 Ph.D. full-time researchers and 10 Ph.D. students
  - Main research lines: sensor networks, optical networks, IVC, PLC, RFID/NFC, P2P VoD...
- Net2Plan team
  - Prof. Pablo Pavón Mariño
  - José Luis Izquierdo Zaragoza

Motivation

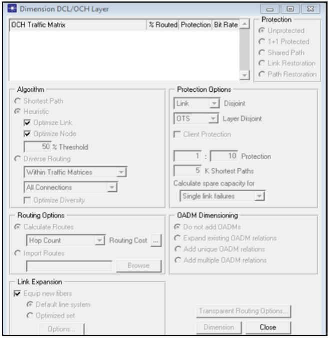
Introduction

- Network planning is in good health: 200+ contributions from 2010 to 2012 only for optical networks planning in IEEEXplore
- Hypothesis: Existing tools are a bottleneck
- Different requirements from users:
  - Research: Fast prototyping/testing, REUSE CODE (open-source algorithm repository), validate, compare results...
  - Industry: Application of research results, and prospective studies
  - Education: Put focus on planning skills (i.e. network optimization, algorithm complexity...)

A day in the life of a Ph.D. student I

Motivation

- Implementing network design algorithms:
  - Alternative 1: *Ad-hoc* toolchain
    1. Generate ILP `.mod` file
    2. Use a topology generator
    3. Use a traffic generator
    4. Convert topology and traffic to `.dat` file
    5. Run ILP solver
    6. Process output data
    7. Plot graphs
  - Alternative 2: Using existing tools?



Dimension DCL/OCH Layer

OCH Traffic Matrix	% Routed	Protection	Bit Rate

Protection
● Unprotected
○ 1+1 Protected
○ Shared Path
○ Link Restoration
○ Path Restoration

Algorithm
○ Shortest Path
● Heuristic
☑ Optimize Link
☑ Optimize Node
50 % Threshold
○ Diverse Routing
Within Traffic Matrices
All Connections
□ Optimize Diversity

Protection Options
Link Disjoint
OTS Layer Disjoint
□ Client Protection
1 : 10 Protection
5 K Shortest Paths
Calculate spare capacity for
Single link failures

Routing Options
● Calculate Routes
Hop Count Routing Cost ...
○ Import Routes
Browse

OADM Dimensioning
● Do not add OADMs
○ Expand existing OADM relations
○ Add unique OADM relations
○ Add multiple OADM relations

Link Expansion
☑ Equip new fibers
● Default line system
○ Optimized set
Options...

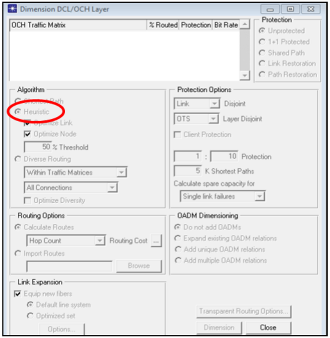
Transparent Routing Options...
Dimension
Close

OPNET SP Guru Transport Planner

A day in the life of a Ph.D. student I

Motivation

- Implementing network design algorithms:
  - Alternative 1: *Ad-hoc* toolchain
    1. Generate ILP `.mod` file
    2. Use a topology generator
    3. Use a traffic generator
    4. Convert topology and traffic to `.dat` file
    5. Run ILP solver
    6. Process output data
    7. Plot graphs
  - Alternative 2: Using existing tools?



Dimension DCL/OCH Layer

OCH Traffic Matrix	% Routed	Protection	Bit Rate

Protection
● Unprotected
○ 1+1 Protected
○ Shared Path
○ Link Restoration
○ Path Restoration

Algorithm
○ Shortest Path
● Heuristic
☑ Optimize Link
☑ Optimize Node
50 % Threshold
○ Diverse Routing
Within Traffic Matrices
All Connections
□ Optimize Diversity

Protection Options
Link Disjoint
OTS Layer Disjoint
□ Client Protection
1 : 10 Protection
5 K Shortest Paths
Calculate spare capacity for
Single link failures

Routing Options
● Calculate Routes
Hop Count Routing Cost
○ Import Routes

OADM Dimensioning
● Do not add OADMs
○ Expand existing OADM relations
○ Add unique OADM relations
○ Add multiple OADM relations

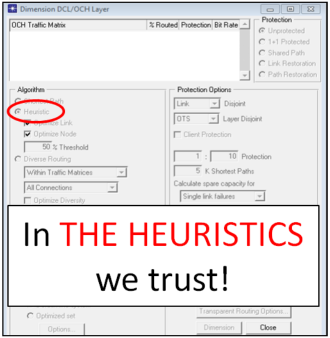
Link Expansion
☑ Equip new fibers
● Default line system
○ Optimized set

OPNET SP Guru Transport Planner

A day in the life of a Ph.D. student I

Motivation

- Implementing network design algorithms:
  - Alternative 1: *Ad-hoc* toolchain
    1. Generate ILP `.mod` file
    2. Use a topology generator
    3. Use a traffic generator
    4. Convert topology and traffic to `.dat` file
    5. Run ILP solver
    6. Process output data
    7. Plot graphs
  - Alternative 2: Using existing tools?



Dimension DCL/OCH Layer

OCH Traffic Matrix	% Routed	Protection	Bit Rate

Protection
● Unprotected
○ 1+1 Protected
○ Shared Path
○ Link Restoration
○ Path Restoration

Algorithm
○ Shortest Path
● Heuristic
☑ Optimize Link
☑ Optimize Node
50 % Threshold
○ Diverse Routing
Within Traffic Matrices
All Connections
□ Optimize Diversity

Protection Options
Link Disjoint
OTS Layer Disjoint
□ Client Protection
1 : 10 Protection
5 K Shortest Paths
Calculate spare capacity for
Single link failures

○ Optimized set

In THE HEURISTICS we trust!

OPNET SP Guru Transport Planner

A day in the life of a Ph.D. student II

Motivation

- What if we would like to...
  - ...test a new protection/restoration/CAC/traffic anomaly reaction algorithm?
  - ...repeat results from the paper presented in...?
  - ...make prospective studies for a non-mature technology?
- Typical answer: Make it from the scratch
- Consequences: (a lot of) waste of time, frustration, desmotivation...

Outline

1. Introduction
   - About us
   - Motivation
2. Net2Plan
   - Features
   - Tools
   - Website
3. Modeling network designs
4. What's next?

Net2Plan

- Our solution: Net2Plan – A publicly-available Java-based OPEN-SOURCE (LGPL license) network planner
- Allows to integrate user-made algorithms and perform several simulation studies in a technology-agnostic environment

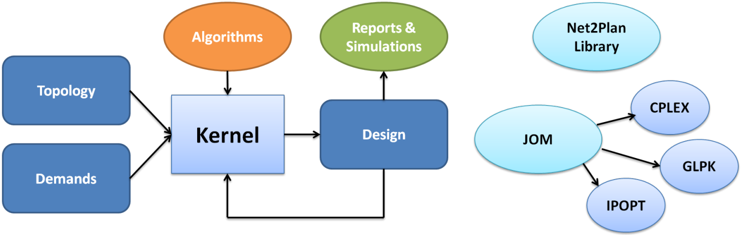
```
public String executeAlgorithm(NetPlan netPlan,
    Map<String, String> algorithmParameters,
    Map<String, String> net2planParameters)
{
    /**
     * Your code here
     */

    return "It works!";
}
```

- History:
  - Origins: September 2011
  - First version: 0.1.0 - February 2012 (MATLAB)
  - Current version: 0.2.2 - October 2013 (Java)

Features

Net2Plan

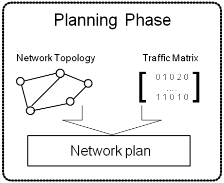


- Website: `http://www.net2plan.com`
  - Download
  - User's guide and Library API Javadoc
  - Teaching materials
  - Examples
- JOM library: `http://ait.upct.es/~ppavon/jom/`

Tools I

Net2Plan

- Network design:



Source: Varvarigos *et al.*, OFC 2013

- Traffic matrix design:

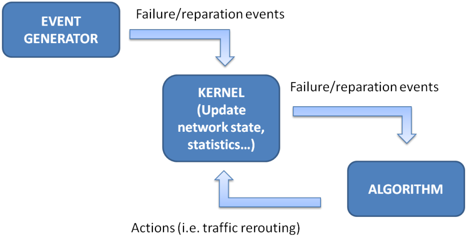
$$\begin{pmatrix} \gamma_{11} & \gamma_{12} & \cdots & \gamma_{1i} \\ \gamma_{21} & \gamma_{22} & & \vdots \\ \vdots & & \ddots & \\ \gamma_{i1} & \cdots & & \gamma_{ij} \end{pmatrix}$$

$$\gamma_{ij} = \frac{Level(L_i, L_j) \cdot (1 - rf + 2 \cdot rf \cdot rand()) \cdot \left(\frac{Pop_i \cdot Pop_j}{Pop_{max}^2} + Pop_{off}\right)^{Pop_{Power}}}{\left(\frac{dist(i,j)}{dist_{max}} + dist_{off}\right)^{Dist_{Power}}}$$

$$0 \leq rf \leq 1 \begin{cases} rf = 0 \Rightarrow (1 - rf + 2 \cdot rf \cdot rand()) = 1 \Rightarrow \text{without random component} \\ rf = 1 \Rightarrow (1 - rf + 2 \cdot rf \cdot rand()) = 2 \cdot rand() = 0 \ldots 2 \end{cases}$$

Tools II
Net2Plan

- Resilience simulation (discrete-event simulation):

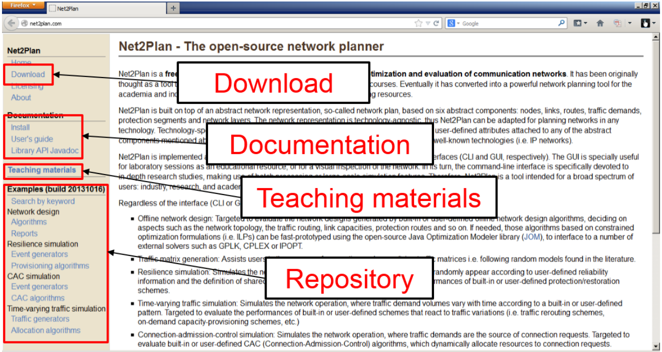

EVENT GENERATOR
Failure/reparation events
KERNEL (Update network state, statistics...)
Failure/reparation events
ALGORITHM
Actions (i.e. traffic rerouting)

- Connection-admission-control simulation (discrete-event simulation): where events are connection requests/departures
- Time-varying traffic simulation (discrete-event simulation): where events are variations on traffic demand volume

Website I

Net2Plan

- http://www.net2plan.com/



Net2Plan - The open-source network planner

Net2Plan is a **free** … **timization and evaluation of communication networks**. It has been originally thought as a tool … courses. Eventually it has converted into a powerful network planning tool for the academia and ind… g resources.

Net2Plan is built on top of an abstract network representation, so-called network plan, based on six abstract components: nodes, links, routes, traffic demands, protection segments and network layers. The network representation is technology-agnostic, thus Net2Plan can be adapted for planning networks in any technology. Technology-sp… user-defined attributes attached to any of the abstract components mentioned ab… well-known technologies (i.e. IP networks).

Net2Plan is implemented … erfaces (CLI and GUI, respectively). The GUI is specially useful for laboratory sessions as an educational resource, or for a visual inspection of the network. In its turn, the command-line interface is specifically devoted to in depth research studies, making us… is a tool intended for a broad spectrum of users: industry, research, and acader…

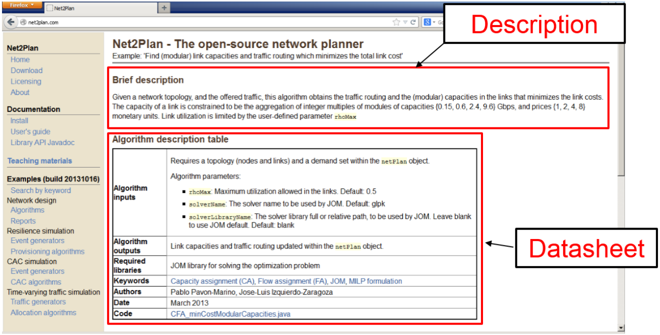
Regardless of the interface (CLI or G…

- Offline network design: Targeted … network design algorithms, deciding on aspects such as the network topology, the traffic routing, link capacities, protection routes and so on. If needed, those algorithms based on constrained optimization formulations (i.e. ILPs) can be fast-prototyped using the open-source Java Optimization Modeler library (JOM), to interface to a number of external solvers such as GPLK, CPLEX or IPOPT.
- Traffic matrix generation: Assists users … matrices i.e. following random models found in the literature.
- Resilience simulation: Simulates the n… randomly appear according to user-defined reliability information and the definition of share… rmances of built-in or user-defined protection/restoration schemes.
- Time-varying traffic simulation: Simulates the network operation, where traffic demand volumes vary with time according to a built-in or user-defined pattern. Targeted to evaluate the performances of built-in or user-defined schemes that react to traffic variations (i.e. traffic rerouting schemes, on-demand capacity-provisioning schemes, etc.)
- Connection-admission-control simulation: Simulates the network operation, where traffic demands are the source of connection requests. Targeted to evaluate built-in or user-defined CAC (Connection-Admission-Control) algorithms, which dynamically allocate resources to connection requests.

Website II

Net2Plan

- Example from the repository:



Description

Net2Plan - The open-source network planner

Example: 'Find (modular) link capacities and traffic routing which minimizes the total link cost'

Brief description

Given a network topology, and the offered traffic, this algorithm obtains the traffic routing and the (modular) capacities in the links that minimizes the link costs. The capacity of a link is constrained to be the aggregation of integer multiples of modules of capacities {0.15, 0.6, 2.4, 9.6} Gbps, and prices {1, 2, 4, 8} monetary units. Link utilization is limited by the user-defined parameter `rhoMax`

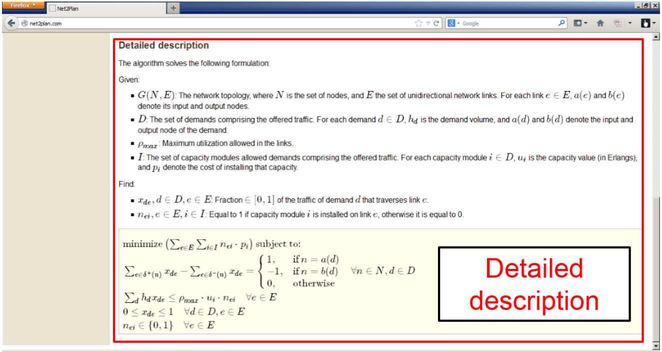
Algorithm description table

Algorithm inputs	Requires a topology (nodes and links) and a demand set within the `netPlan` object. Algorithm parameters: ▪ `rhoMax`: Maximum utilization allowed in the links. Default: 0.5 ▪ `solverName`: The solver name to be used by JOM. Default: glpk ▪ `solverLibraryName`: The solver library full or relative path, to be used by JOM. Leave blank to use JOM default. Default: blank
Algorithm outputs	Link capacities and traffic routing updated within the `netPlan` object.
Required libraries	JOM library for solving the optimization problem
Keywords	Capacity assignment (CA), Flow assignment (FA), JOM, MILP formulation
Authors	Pablo Pavon-Marino, Jose-Luis Izquierdo-Zaragoza
Date	March 2013
Code	CFA_minCostModularCapacities.java

Datasheet

Website III
Net2Plan

- Example from the repository:



Detailed description

The algorithm solves the following formulation:

Given:

- $G(N,E)$: The network topology, where N is the set of nodes, and E the set of unidirectional network links. For each link $e \in E$, $a(e)$ and $b(e)$ denote its input and output nodes.
- D: The set of demands comprising the offered traffic. For each demand $d \in D$, h_d is the demand volume, and $a(d)$ and $b(d)$ denote the input and output node of the demand.
- ρ_{max}: Maximum utilization allowed in the links.
- I: The set of capacity modules allowed demands comprising the offered traffic. For each capacity module $i \in D$, u_i is the capacity value (in Erlangs), and p_i denote the cost of installing that capacity.

Find:

- $x_{de}, d \in D, e \in E$: Fraction $\in [0,1]$ of the traffic of demand d that traverses link e.
- $n_{ei}, e \in E, i \in I$: Equal to 1 if capacity module i is installed on link e, otherwise it is equal to 0.

$$\text{minimize } \left(\textstyle\sum_{e \in E} \sum_{i \in I} n_{ei} \cdot p_i\right) \text{ subject to:}$$

$$\textstyle\sum_{e \in \delta^+(n)} x_{de} - \sum_{e \in \delta^-(n)} x_{de} = \begin{cases} 1, & \text{if } n = a(d) \\ -1, & \text{if } n = b(d) \\ 0, & \text{otherwise} \end{cases} \quad \forall n \in N, d \in D$$

$$\textstyle\sum_d h_d x_{de} \leq \rho_{max} \cdot u_i \cdot n_{ei} \quad \forall e \in E$$

$$0 \leq x_{de} \leq 1 \quad \forall d \in D, e \in E$$

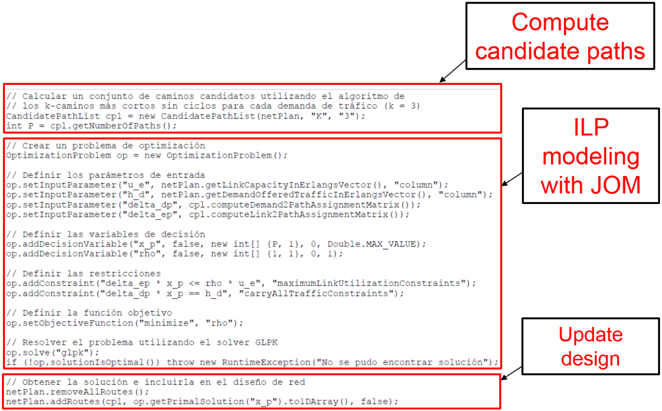
$$n_{ei} \in \{0,1\} \quad \forall e \in E$$

Detailed description

Website IV

Net2Plan

- Example from the repository:



Compute candidate paths

ILP modeling with JOM

Update design

```
// Calcular un conjunto de caminos candidatos utilizando el algoritmo de
// los k-caminos más cortos sin ciclos para cada demanda de tráfico (k = 3)
CandidatePathList cpl = new CandidatePathList(netPlan, "K", "3");
int P = cpl.getNumberOfPaths();

// Crear un problema de optimización
OptimizationProblem op = new OptimizationProblem();

// Definir los parámetros de entrada
op.setInputParameter("u_e", netPlan.getLinkCapacityInErlangsVector(), "column");
op.setInputParameter("h_d", netPlan.getDemandOfferedTrafficInErlangsVector(), "column");
op.setInputParameter("delta_dp", cpl.computeDemand2PathAssignmentMatrix());
op.setInputParameter("delta_ep", cpl.computeLink2PathAssignmentMatrix());

// Definir las variables de decisión
op.addDecisionVariable("x_p", false, new int[] {P, 1}, 0, Double.MAX_VALUE);
op.addDecisionVariable("rho", false, new int[] {1, 1}, 0, 1);

// Definir las restricciones
op.addConstraint("delta_ep * x_p <= rho * u_e", "maximumLinkUtilizationConstraints");
op.addConstraint("delta_dp * x_p == h_d", "carryAllTrafficConstraints");

// Definir la función objetivo
op.setObjectiveFunction("minimize", "rho");

// Resolver el problema utilizando el solver GLPK
op.solve("glpk");
if (!op.solutionIsOptimal()) throw new RuntimeException("No se pudo encontrar solución");

// Obtener la solución e incluirla en el diseño de red
netPlan.removeAllRoutes();
netPlan.addRoutes(cpl, op.getPrimalSolution("x_p").to1DArray(), false);
```

Outline

1 Introduction
   - About us
   - Motivation

2 Net2Plan
   - Features
   - Tools
   - Website

3 Modeling network designs

4 What's next?

Modeling network designs

- Idea: Network model as flexible as possible
- Technology-agnostic model based on abstract concepts with a minimum set of member variables:
  - Nodes
  - Links
  - Demands
  - Routes
  - Protection segments
- Users can extend the model (i.e. to make it technology-specific) via key-value additional attributes

Physical topology

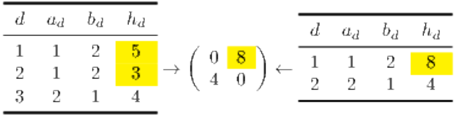
Modeling network designs

- Set of nodes N
  - id: unique identifier
  - position: (x, y) in a 2D plane
  - name: node name
- Set of unidirectional links E (self-links are forbidden)
  - id: unique identifier
  - origin node: id of the origin node
  - destination node: id of the destination node
  - capacity (in Erlangs)
  - length (in Km)

Traffic model

Modeling network designs

- Set of unidirectional (and unicast) demands *D*
    - id: unique identifier
    - ingress node: id of the ingress node
    - egress node: id of the egress node
    - offered traffic (in Erlangs)
- Why using traffic matrices is not recommended?



d	a_d	b_d	h_d
1	1	2	5
2	1	2	3
3	2	1	4

$$\rightarrow \begin{pmatrix} 0 & 8 \\ 4 & 0 \end{pmatrix} \leftarrow$$

d	a_d	b_d	h_d
1	1	2	8
2	2	1	4

Routing model
Modeling network designs

- Set of unidirectional routes (paths) P
  - id: unique identifier
  - demand: id of the associated demand
  - carried traffic (in Erlangs)
  - sequence of links: ids of the corresponding links
  - backup segment list: ids of the backup segments
- Other (bad) alternatives:
  - Demand-link routing: post-processing of flow-formulation variables
  - Destination-based routing: no QoS
- Available routing schemes:
  - Bifurcated/non-bifurcated routing
  - Integral routing

Protection model model I

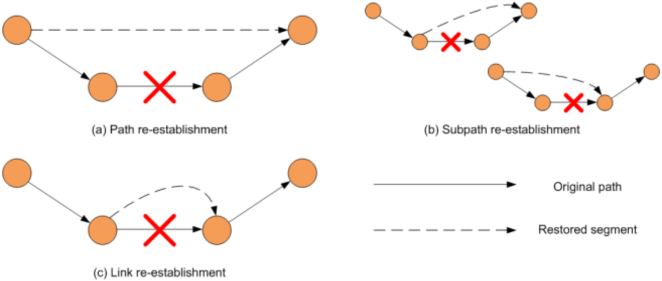
Modeling network designs

- Set of protection segments S
  - id: unique identifier
  - origin node: id of the origin node
  - destination node: id of the destination node
  - reserved bandwidth (in Erlangs)
  - sequence of links: ids of the corresponding links
- Available protection schemes:
  - Dedicated protection: Each segment is associated at most to one traffic route
  - Shared protection: Each segment is associated at least to one traffic route
  - Partial protection: If reserved bandwidth is below the carried one

Protection model model II

Modeling network designs

- Path/sub-path/link protection



(a) Path re-establishment

(b) Subpath re-establishment

(c) Link re-establishment

Notation

Modeling network designs

Element	Parameter	Description
Nodes	N	Set of nodes $n \in N$
	$\delta^+(n), \delta^-(n)$	Set of outgoing and incoming links from/to node n
Links	E	Set of links $e \in E$
	$a(e), b(e)$	Origin and destination nodes of link e
	l_e	Length of link e (Km)
	u_e	Capacity of link e (Erlangs)
	$\mathbf{u}$	Vector form of u_e
	y_e	Traffic carried by link e (Erlangs)
	$\mathbf{y}$	Vector form of y_e
Demands	D	Set of demands $d \in D$
	$a(d), b(d)$	Ingress and egress nodes of demand d
	h_d	Offered traffic for demand d
	$\mathbf{h}$	Vector form of h_d
	r_d	Carried traffic for demand d
	$\mathbf{r}$	Vector form of r_d
Routing	P	Set of paths $p \in P$
	$P_d \subseteq P$	Subset of the paths in P that are associated to demand d
	$P_e \subseteq P$	Subset of the paths in P that traverse link e
	x_p	Traffic volume carried by path p
	$\mathbf{x}$	Vector form of x_p
	$a(p), b(p), l(p)$	Origin and destination nodes, and number of hops of path p
	$d(p)$	Demand corresponding to path p
Protection segments	S	Set of protection segments $(s \in S)$
	$S_e \subseteq S$	Subset of the protection segments in S that traverse link e
	$S_p \subseteq S$	Subset of the protection segments in S that are associated to path p
	$a(s), b(s), l(s)$	Origin and destination nodes, and number of hops of protection segment s
	u_s	Reserved bandwidth for protection segment s (Erlangs)

Outline

1. Introduction
   - About us
   - Motivation
2. Net2Plan
   - Features
   - Tools
   - Website
3. Modeling network designs
4. What's next?

What's next?

- Review of some (classical) problems: Live demo
- Lab work:
  - ILPs for RWA
  - Heuristic for RFWA

  $\Big\}$ Communication Network Design (Prof. Tornatore)
  - Column generation ← Graph Optimization (Prof. Carello)
- Real-world case study: IP-over-WDM

Universidad Politécnica de Cartagena

git Ingeniería Telemática

Thank you for your attention!!! Questions?

José Luis Izquierdo Zaragoza
http://ait.upct.es/~jlizquierdo
josel.izquierdo@upct.es

Net2Plan: The open-source network planner

José Luis Izquierdo Zaragoza
josel.izquierdo@upct.es
`http://ait.upct.es/~jlizquierdo`

Politecnico di Milano

Milano, Italy, November 22-27, 2013

This research was partially supported by a predoctoral FPU fellowship from the Spanish Ministry of Education, Culture and Sport (reference no. FPU12/04571) and a mobility grant from Universidad Politécnica de Cartagena (PMPDI-UPCT-2013)