gnpy Documentation

Telecom Infra Project - OOPT PSE Group

Jun 06, 2021

CONTENTS

GNPy is an open-source, community-developed library for building route planning and optimization tools in real-world mesh optical networks. It is based on the Gaussian Noise Model.

CHAPTER

ONE

INTRODUCTION

`gnpy` is a library for building route planning and optimization tools.

It ships with a number of example programs. Release versions will ship with fully-functional programs.

> **Note**: *If you are a network operator or involved in route planning and optimization for your organization, please contact project maintainer Jan Kundrát <jan.kundrat@telecominfraproject.com>. gnpy is looking for users with specific, delineated use cases to drive requirements for future development.*

This example demonstrates how GNPy can be used to check the expected SNR at the end of the line by varying the channel input power:

By default, this script operates on a single span network defined in gnpy/example-data/edfa_example_network.json

You can specify a different network at the command line as follows. For example, to use the CORONET Global network defined in gnpy/example-data/CORONET_Global_Topology.json:

```
$ gnpy-transmission-example $(gnpy-example-data)/CORONET_Global_Topology.json
```

It is also possible to use an Excel file input (for example gnpy/example-data/CORONET_Global_Topology.xls). The Excel file will be processed into a JSON file with the same prefix. Further details about the Excel data structure are available in the documentation.

The main transmission example will calculate the average signal OSNR and SNR across network elements (transceiver, ROADMs, fibers, and amplifiers) between two transceivers selected by the user. Additional details are provided by doing `gnpy-transmission-example -h`. (By default, for the CORONET Global network, it will show the transmission of spectral information between Abilene and Albany)

This script calculates the average signal OSNR = P_{ch}/P_{ase} and SNR = $P_{ch}/(P_{nli}+P_{ase})$.

P_{ase} is the amplified spontaneous emission noise, and P_{nli} the non-linear interference noise.

1.1 Further Instructions for Use

Simulations are driven by a set of JSON or XLS files.

The `gnpy-transmission-example` script propagates a spectrum of channels at 32 Gbaud, 50 GHz spacing and 0 dBm/channel. Launch power can be overridden by using the `--power` argument. Spectrum information is not yet parametrized but can be modified directly in the `eqpt_config.json` (via the `SpectralInformation` -SI-structure) to accommodate any baud rate or spacing. The number of channel is computed based on `spacing` and `f_min`, `f_max` values.

An experimental support for Raman amplification is available:

```
$ gnpy-transmission-example \
  $(gnpy-example-data)/raman_edfa_example_network.json \
  --sim $(gnpy-example-data)/sim_params.json --show-channels
```

Configuration of Raman pumps (their frequencies, power and pumping direction) is done via the RamanFiber element in the network topology. General numeric parameters for simulation control are provided in the gnpy/example-data/sim_params.json.

Use `gnpy-path-request` to request several paths at once:

```
$ cd $(gnpy-example-data)
$ gnpy-path-request -o output_file.json \
  meshTopologyExampleV2.xls meshTopologyExampleV2_services.json
```

This program operates on a network topology (JSON or Excel format), processing the list of service requests (JSON or XLS again). The service requests and reply formats are based on the draft-ietf-teas-yang-path-computation-01 with custom extensions (e.g., for transponder modes). An example of the JSON input is provided in file *service-template.json*, while results are shown in *path_result_template.json*.

Important note: `gnpy-path-request` is not a network dimensionning tool: each service does not reserve spectrum, or occupy ressources such as transponders. It only computes path feasibility assuming the spectrum (between defined frequencies) is loaded with "nb of channels" spaced by "spacing" values as specified in the system parameters input in the service file, each cannel having the same characteristics in terms of baudrate, format,... as the service transponder. The transceiver element acts as a "logical starting/stopping point" for the spectral information propagation. At that point it is not meant to represent the capacity of add drop ports. As a result transponder type is not part of the network info. it is related to the list of services requests.

The current version includes a spectrum assigment features that enables to compute a candidate spectrum assignment for each service based on a first fit policy. Spectrum is assigned based on service specified spacing value, path_bandwidth value and selected mode for the transceiver. This spectrum assignment includes a basic capacity planning capability so that the spectrum resource is limited by the frequency min and max values defined for the links. If the requested services reach the link spectrum capacity, additional services feasibility are computed but marked as blocked due to spectrum reason.

OpenROADM networks can be simulated via `gnpy/example-data/eqpt_config_openroadm.json` – see `gnpy/example-data/Sweden_OpenROADM_example_network.json` as an example.

CHAPTER

TWO

SIMULATING NETWORKS WITH GNPY

Running simulations with GNPy requires three pieces of information:

- the *network topology*, which describes how the network looks like, what are the fiber lengths, what amplifiers are used, etc.,
- the *equipment library*, which holds machine-readable datasheets of the equipment used in the network,
- the *simulation options* holding instructions about what to simulate, and under which conditions.

2.1 Network Topology

The *topology* acts as a "digital self" of the simulated network. When given a network topology, GNPy can either run a specific simulation as-is, or it can *optimize* the topology before performing the simulation.

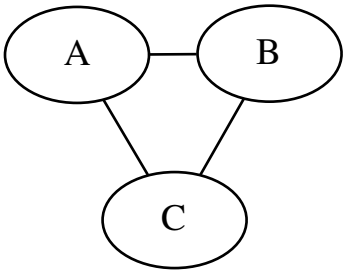
A network topology for GNPy is often a generic, mesh network. This enables GNPy to take into consideration the current spectrum allocation as well as availability and resiliency considerations. When the time comes to run a particular *propagation* of a signal and its impairments are computed, though, a linear path through the network is used. For this purpose, the *path* through the network refers to an ordered, acyclic sequence of *nodes* that are processed. This path is directional, and all "GNPy elements" along the path match the unidirectional part of a real-world network equipment.

Note: In practical terms, an amplifier in GNPy refers to an entity with a single input port and a single output port. A real-world inline EDFA enclosed in a single chassis will be therefore represented as two GNPy-level amplifiers.

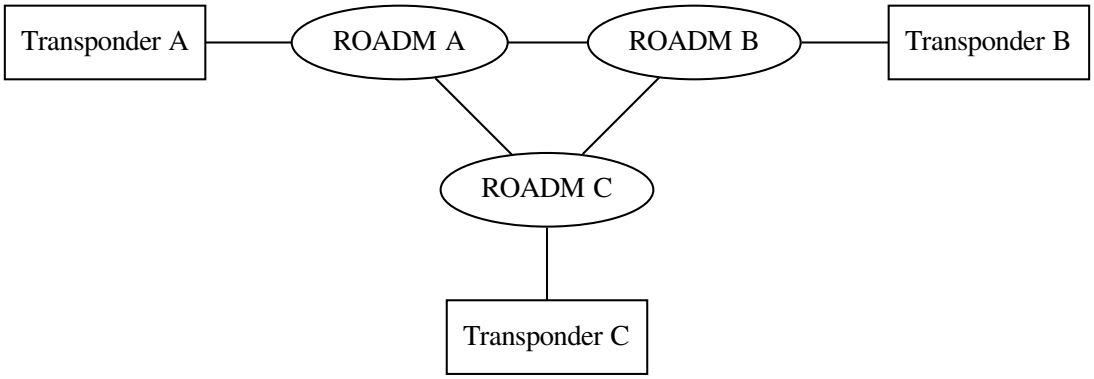
The network topology contains not just the physical topology of the network, but also references to the *equipment library* and a set of *operating parameters* for each entity. These parameters include the **fiber length** of each fiber, the connector **attenutation losses**, or an amplifier's specific **gain setting**.

2.1.1 Fully Specified vs. Partially Designed Networks

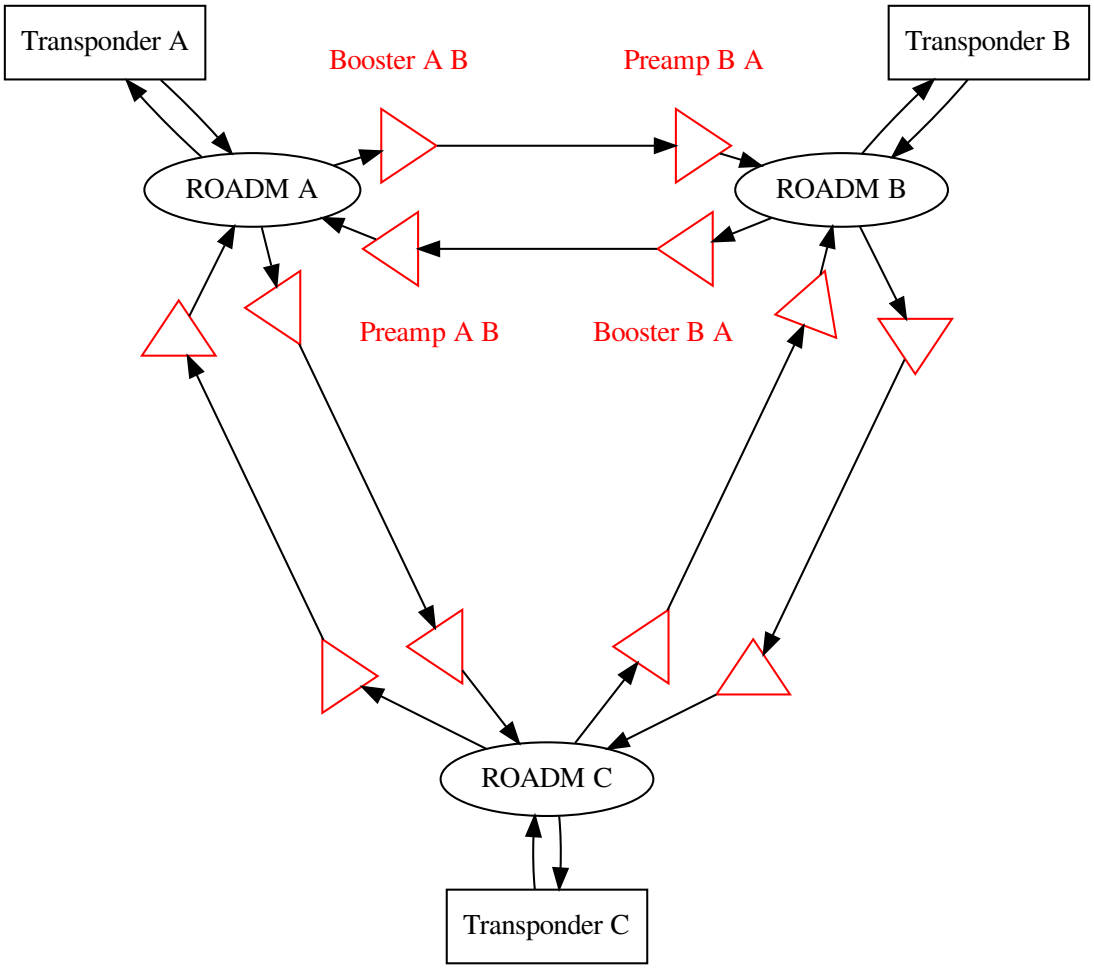
Let's consider a simple triangle topology with three PoPs (Points of Presence) covering three cities:



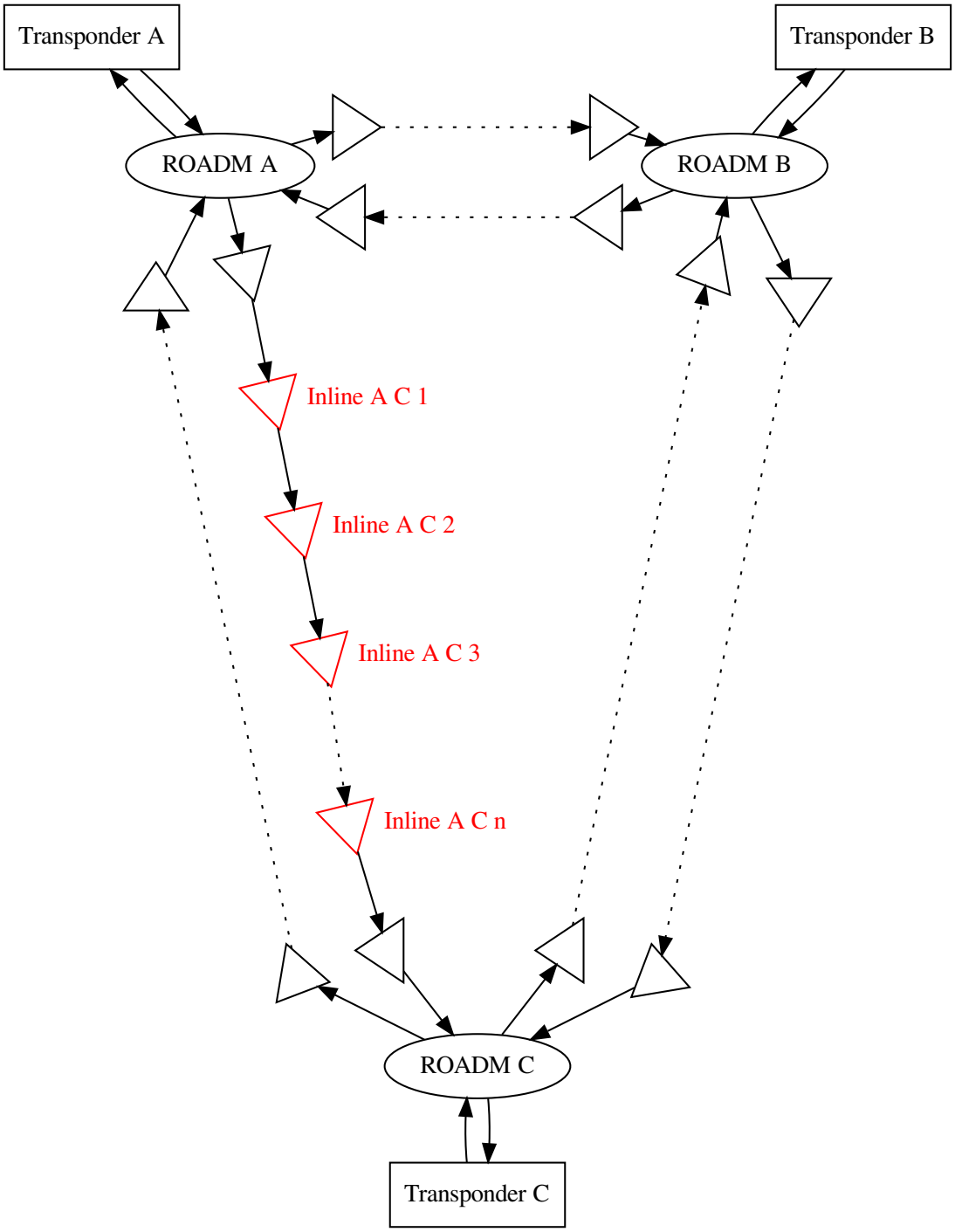
In the real world, each city would probably host a ROADM and some transponders:



GNPy simulation works by propagating the optical signal over a sequence of elements, which means that one has to add some preamplifiers and boosters. The amplifiers are, by definition, unidirectional, so the graph becomes quite complex:



In many regions, the ROADMs are not placed physically close to each other, so the long-haul fiber links (OMS (Optical Multiplex Section)) are split into individual spans (OTS (Optical Transport Section)) by in-line amplifiers, resulting in an even more complicated topology graphs:



In such networks, GNPy's autodesign features becomes very useful. It is possible to connect ROADMs via "tentative links" which will be replaced by a sequence of actual fibers and specific amplifiers. In other cases where the location of amplifier huts is already known, but the specific EDFA models have not yet been decided, one can put in amplifier placeholders and let GNPy assign the best amplifier.

2.2 The Equipment Library

In order to produce an accurate simulation, GNPy needs to know the physical properties of each entity which affects the optical signal. Entries in the equipment library correspond to actual real-world, tangible entities. Unlike a typical NMS (Network Management System), GNPy considers not just the active NEs (Network Elements) such as amplifiers and ROADMs (Reconfigurable Optical Add/Drop Multiplexers), but also the passive ones, such as the optical fiber.

As the signal propagates through the network, the largest source of optical impairments is the noise introduced from amplifiers. An accurate description of the EDFA (Erbium-Doped Fiber Amplifier) and especially its noise characteristics is required. GNPy describes this property in terms of the **Noise Figure (NF)** of an amplifier model as a function of its operating point.

The amplifiers compensate power losses induced on the signal in the optical fiber. The linear losses, however, are just one phenomenon of a multitude of effects that affect the signals in a long fiber run. While a more detailed description is available *in the literature*, for the purpose of the equipment library, the description of the *optical fiber* comprises its **linear attenutation coefficient**, a set of parameters for the **Raman effect**, optical **dispersion**, etc.

Signals are introduced into the network via *transponders*. The set of parameters that are required describe the physical properties of each supported *mode* of the transponder, including its **symbol rate**, spectral **width**, etc.

In the junctions of the network, *ROADMs* are used for spectrum routing. GNPy currently does not take into consideration the spectrum filtering penalties of the WSSEs (Wavelength Selective Switches), but the equipment library nonetheless contains a list of required parameters, such as the attenuation options, so that the network can be properly simulated.

2.2.1 Amplifier Noise Figure Models

One of the key parameters of an amplifier is the method to use for computing the Noise Figure (NF). GNPy supports several different noise models with varying level of accuracy. When in doubt, contact your vendor's technical support and ask them to *contribute their equipment descriptions* to GNPy.

The most accurate noise models describe the resulting NF of an EDFA as a third-degree polynomial. GNPy understands polynomials as a NF-yielding function of the *gain difference from the optimal gain*, or as a function of the input power resulting in an incremental OSNR as used in *OpenROADM inline amplifiers* and *OpenROADM booster/preamps in the ROADMs*. For scenarios where the vendor has not yet contributed an accurate EDFA NF description to GNPy, it is possible to approximate the characteristics via an operator-focused, min-max NF model.

2.2.1.1 Min-max NF

This is an operator-focused model where performance is defined by the *minimal* and *maximal NF*. These are especially suited to model a dual-coil EDFA with a VOA in between. In these amplifiers, the minimal NF is achieved when the EDFA operates at its maximal (and usually optimal, in terms of flatness) gain. The worst (maximal) NF applies when the EDFA operates at its minimal gain.

This model is suitable for use when the vendor has not provided a more accurate performance description of the EDFA.

2.2.1.2 Raman Approximation

While GNPy is fully Raman-aware, under certain scenarios it is useful to be able to run a simulation without an accurate Raman description. For these purposes the *polynomial NF* model with a $=$ b $=$ c $= 0$, and d $= NF$ can be used.

2.3 Simulation

When the network model has been instantiated and the physical properties and operational settings of the actual physical devices are known, GNPy can start simulating how the signal propagate through the optical fiber.

This set of input parameters include options such as the *spectrum allocation*, i.e., the number of channels and their spacing. Various strategies for network optimization can be provided as well.

CHAPTER

THREE

INSTALLING GNPY

There are several methods on how to obtain GNPy. The easiest option for a non-developer is probably going via our *Docker images*. Developers are encouraged to install the *Python package in the same way as any other Python package*. Note that this needs a *working installation of Python*, for example *via Anaconda*.

3.1 Using prebuilt Docker images

Our Docker images contain everything needed to run all examples from this guide. Docker transparently fetches the image over the network upon first use. On Linux and Mac, run:

```
$ docker run -it --rm --volume $(pwd):/shared telecominfraproject/oopt-gnpy
root@bea050f186f7:/shared/example-data#
```

On Windows, launch from Powershell as:

```
PS C:\> docker run -it --rm --volume ${PWD}:/shared telecominfraproject/oopt-gnpy
root@89784e577d44:/shared/example-data#
```

In both cases, a directory named `example-data/` will appear in your current working directory. GNPy automaticallly populates it with example files from the current release. Remove that directory if you want to start from scratch.

3.2 Using Python on your computer

> **Note**: *gnpy* supports Python 3 only. Python 2 is not supported. *gnpy* requires Python 3.6
>
> **Note**: the *gnpy* maintainers strongly recommend the use of Anaconda for managing dependencies.

It is recommended that you use a "virtual environment" when installing *gnpy*. Do not install *gnpy* on your system Python.

We recommend the use of the Anaconda Python distribution which comes with many scientific computing dependencies pre-installed. Anaconda creates a base "virtual environment" for you automatically. You can also create and manage your `conda` "virtual environments" yourself (see: https://conda.io/docs/user-guide/tasks/manage-environments.html)

To activate your Anaconda virtual environment, you may need to do the following:

```
$ source /path/to/anaconda/bin/activate # activate Anaconda base environment
(base) $                                # note the change to the prompt
```

You can check which Anaconda environment you are using with:

```
(base) $ conda env list                          # list all environments
# conda environments:
#
base                  *  /src/install/anaconda3

(base) $ echo $CONDA_DEFAULT_ENV                 # show default environment
base
```

You can check your version of Python with the following. If you are using Anaconda's Python 3, you should see similar output as below. Your results may be slightly different depending on your Anaconda installation path and the exact version of Python you are using.

```
$ which python                   # check which Python executable is used
/path/to/anaconda/bin/python
$ python -V                      # check your Python version
Python 3.6.5 :: Anaconda, Inc.
```

3.2.1 Installing the Python package

From within your Anaconda Python 3 environment, you can clone the master branch of the *gnpy* repo and install it with:

```
$ git clone https://github.com/Telecominfraproject/oopt-gnpy # clone the repo
$ cd oopt-gnpy
$ pip install --editable . # note the trailing dot
```

To test that *gnpy* was successfully installed, you can run this command. If it executes without a `ModuleNotFoundError`, you have successfully installed *gnpy*.

```
$ python -c 'import gnpy' # attempt to import gnpy

$ pytest                  # run tests
```

CHAPTER

FOUR

JSON INPUT FILES

GNPy uses a set of JSON files for modeling the network. Some data (such as network topology or the service requests) can be also passed via *XLS files*.

4.1 Equipment Library

Design and transmission parameters are defined in a dedicated json file. By default, this information is read from gnpy/example-data/eqpt_config.json. This file defines the equipment libraries that can be customized (EDFAs, fibers, and transceivers).

It also defines the simulation parameters (spans, ROADMs, and the spectral information to transmit.)

4.1.1 EDFA

The EDFA equipment library is a list of supported amplifiers. New amplifiers can be added and existing ones removed. Three different noise models are available:

1. `'type_def': 'variable_gain'` is a simplified model simulating a 2-coil EDFA with internal, input and output VOAs. The NF vs gain response is calculated accordingly based on the input parameters: `nf_min`, `nf_max`, and `gain_flatmax`. It is not a simple interpolation but a 2-stage NF calculation.
2. `'type_def': 'fixed_gain'` is a fixed gain model. *NF == Cte == nf0* if *gain_min < gain < gain_flatmax*
3. `'type_def': 'openroadm'` models the incremental OSNR contribution as a function of input power. It is suitable for inline amplifiers that conform to the OpenROADM specification. The input parameters are coefficients of the *third-degree polynomial*.
4. `'type_def': 'openroadm_preamp'` and `openroadm_booster` approximate the *preamp and booster within an OpenROADM network*. No extra parameters specific to the NF model are accepted.
5. `'type_def': 'advanced_model'` is an advanced model. A detailed JSON configuration file is required (by default gnpy/example-data/std_medium_gain_advanced_config.json). It uses a 3rd order polynomial where NF = f(gain), NF_ripple = f(frequency), gain_ripple = f(frequency), N-array dgt = f(frequency). Compared to the previous models, NF ripple and gain ripple are modelled.

For all amplifier models:

field	type	description
`type_variety`	(string)	a unique name to ID the amplifier in the JSON/Excel template topology input file
`out_voa_auto`	(boolean)	auto_design feature to optimize the amplifier output VOA. If true, output VOA is present and will be used to push amplifier gain to its maximum, within EOL power margins.
`allowed_for_design`	(boolean)	If false, the amplifier will not be picked by auto-design but it can still be used as a manual input (from JSON or Excel template topology files.)

4.1.2 Fiber

The fiber library currently describes SSMF and NZDF but additional fiber types can be entered by the user following the same model:

field	type	description
`type_variety`	(string)	a unique name to ID the fiber in the JSON or Excel template topology input file
`dispersion`	(number)	In $s \times m^{-1} \times m^{-1}$.
`dispersion_slope`	(number)	In $s \times m^{-1} \times m^{-1} \times m^{-1}$
`gamma`	(number)	$2\pi \times n^2/(\lambda * A_{eff})$, in $w^{-1} \times m^{-1}$.
`pmd_coef`	(number)	Polarization mode dispersion (PMD) coefficient. In $s \times \sqrt{m}^{-1}$.

4.1.3 Transceiver

The transceiver equipment library is a list of supported transceivers. New transceivers can be added and existing ones removed at will by the user. It is used to determine the service list path feasibility when running the `gnpy-path-request` script.

field	type	description
`type_variety`	(string)	A unique name to ID the transceiver in the JSON or Excel template topology input file
`frequency`	(number)	Min/max central channel frequency.
`mode`	(number)	A list of modes supported by the transponder. New modes can be added at will by the user. The modes are specific to each transponder type_variety. Each mode is described as below.

The modes are defined as follows:

field	type	description
`format`	(string)	a unique name to ID the mode
`baud_rate`	(number)	in Hz
`OSNR`	(number)	min required OSNR in 0.1nm (dB)
`bit_rate`	(number)	in bit/s
`roll_off`	(number)	Pure number between 0 and 1. TX signal roll-off shape. Used by Raman-aware simulation code.
`tx_osnr`	(number)	In dB. OSNR out from transponder.
`cost`	(number)	Arbitrary unit

4.1.4 ROADM

The user can only modify the value of existing parameters:

field	type	description
`target_pch_out_db`	(number)	Auto-design sets the ROADM egress channel power. This reflects typical control loop algorithms that adjust ROADM losses to equalize channels (eg coming from different ingress direction or add ports) This is the default value Roadm/params/target_pch_out_db if no value is given in the `Roadm` element in the topology input description. This default value is ignored if a params/target_pch_out_db value is input in the topology for a given ROADM.
`add_drop_osnr`	(number)	OSNR contribution from the add/drop ports
`pmd`	(number)	Polarization mode dispersion (PMD). (s)
`restrictions`	**(dict of** strings)	If non-empty, keys `preamp_variety_list` and `booster_variety_list` represent list of `type_variety` amplifiers which are allowed for auto-design within ROADM's line degrees. If no booster should be placed on a degree, insert a `Fused` node on the degree output.

4.2 Global parameters

The following options are still defined in `eqpt_config.json` for legacy reasons, but they do not correspond to tangible network devices.

Auto-design automatically creates EDFA amplifier network elements when they are missing, after a fiber, or between a ROADM and a fiber. This auto-design functionality can be manually and locally deactivated by introducing a `Fused` network element after a `Fiber` or a `Roadm` that doesn't need amplification. The amplifier is chosen in the EDFA list of the equipment library based on gain, power, and NF criteria. Only the EDFA that are marked `'allowed_for_design': true` are considered.

For amplifiers defined in the topology JSON input but whose `gain = 0` (placeholder), auto-design will set its gain automatically: see `power_mode` in the `Spans` library to find out how the gain is calculated.

4.2.1 Span

Span configuration is not a list (which may change in later releases) and the user can only modify the value of existing parameters:

field	type	description
power_mode	(boolean)	If false, gain mode. Auto-design sets amplifier gain = preceding span loss, unless the amplifier exists and its gain > 0 in the topology input JSON. If true, power mode (recommended for auto-design and power sweep.) Auto-design sets amplifier power according to delta_power_range. If the amplifier exists with gain > 0 in the topology JSON input, then its gain is translated into a power target/channel. Moreover, when performing a power sweep (see `power_range_db` in the SI configuration library) the power sweep is performed w/r/t this power target, regardless of preceding amplifiers power saturation/limitations.
delta_power_range_db	(number)	Auto-design only, power-mode only. Specifies the [min, max, step] power excursion/span. It is a relative power excursion w/r/t the power_dbm + power_range_db (power sweep if applicable) defined in the SI configuration library. This relative power excursion is = 1/3 of the span loss difference with the reference 20 dB span. The 1/3 slope is derived from the GN model equations. For example, a 23 dB span loss will be set to 1 dB more power than a 20 dB span loss. The 20 dB reference spans will *always* be set to power = power_dbm + power_range_db. To configure the same power in all spans, use *[0, 0, 0]*. All spans will be set to power = power_dbm + power_range_db. To configure the same power in all spans and 3 dB more power just for the longest spans: *[0, 3, 3]*. The longest spans are set to power = power_dbm + power_range_db + 3. To configure a 4 dB power range across all spans in 0.5 dB steps: *[-2, 2, 0.5]*. A 17 dB span is set to power = power_dbm + power_range_db - 1, a 20 dB span to power = power_dbm + power_range_db and a 23 dB span to power = power_dbm + power_range_db + 1
max_fiber_loss	(number)	Maximum linear fiber loss for Raman amplification use.
max_length	(number)	Split fiber lengths > max_length. Interest to support high level topologies that do not specify in line amplification sites. For example the CORONET_Global_Topology.xlsx defines links > 1000km between 2 sites: it couldn't be simulated if these links were not split in shorter span lengths.
length_units	"m" "km"	Unit for `max_length`.
max_loss	(number)	Not used in the current code implementation.
padding	(number)	In dB. Min span loss before putting an attenuator before fiber. Attenuator value Fiber.att_in = max(0, padding - span_loss). Padding can be set manually to reach a higher padding value for a given fiber by filling in the Fiber/params/att_in field in the topology json input [1] but if span_loss = length * loss_coef + att_in + con_in + con_out < padding, the specified att_in value will be completed to have span_loss = padding. Therefore it is not possible to set span_loss < padding.
EOL	(number)	All fiber span loss ageing. The value is added to the con_out (fiber output connector). So the design and the path feasibility are performed with span_loss + EOL. EOL cannot be set manually for a given fiber span (workaround is to specify higher `con_out` loss for this fiber).
con_in, con_out	(number)	Default values if Fiber/params/con_in/out is None in the topology input description. This default value is ignored if a Fiber/params/con_in/out value is input in the topology for a given Fiber.

```
{
    "uid": "fiber (A1->A2)",
    "type": "Fiber",
    "type_variety": "SSMF",
    "params":
    {
          "length": 120.0,
          "loss_coef": 0.2,
          "length_units": "km",
          "att_in": 0,
          "con_in": 0,
          "con_out": 0
    }
}
```

4.2.2 SpectralInformation

The user can only modify the value of existing parameters. It defines a spectrum of N identical carriers. While the code libraries allow for different carriers and power levels, the current user parametrization only allows one carrier type and one power/channel definition.

field	type	description
`f_min`, `f_max`	(number)	In Hz. Define spectrum boundaries. Note that due to backward compatibility, the first channel central frequency is placed at $f_{min} + spacing$ and the last one at f_{max}.
`baud_rate`	(number)	In Hz. Simulated baud rate.
`spacing`	(number)	In Hz. Carrier spacing.
`roll_off`	(number)	Pure number between 0 and 1. TX signal roll-off shape. Used by Raman-aware simulation code.
`tx_osnr`	(number)	In dB. OSNR out from transponder.
`power_dbm`	(number)	Reference channel power. In gain mode (see spans/power_mode = false), all gain settings are offset w/r/t this reference power. In power mode, it is the reference power for Spans/delta_power_range_db. For example, if delta_power_range_db = *[0,0,0]*, the same power=power_dbm is launched in every spans. The network design is performed with the power_dbm value: even if a power sweep is defined (see after) the design is not repeated.
`power_range_db`	(number)	Power sweep excursion around power_dbm. It is not the min and max channel power values! The reference power becomes: power_range_db + power_dbm.
`sys_margins`	(number)	In dB. Added margin on min required transceiver OSNR.

CHAPTER

FIVE

EXCEL (XLS, XLSX) INPUT FILES

`gnpy-transmission-example` gives the possibility to use an excel input file instead of a json file. The program then will generate the corresponding json file for you.

The file named 'meshTopologyExampleV2.xls' is an example.

In order to work the excel file MUST contain at least 2 sheets:

- Nodes
- Links

(In progress) The File MAY contain an additional sheet:

- Eqt
- Service

5.1 Nodes sheet

Nodes sheet contains nine columns. Each line represents a 'node' (ROADM site or an in line amplifier site ILA or a Fused):

```
City (Mandatory) ; State ; Country ; Region ; Latitude ; Longitude ; Type
```

- **City** is used for the name of a node of the graph. It accepts letters, numbers,underscore,dash, blank… (not exhaustive). The user may want to avoid commas for future CSV exports.

  City name MUST be unique

- **Type** is not mandatory.
  - If not filled, it will be interpreted as an 'ILA' site if node degree is 2 and as a ROADM otherwise.
  - If filled, it can take "ROADM", "FUSED" or "ILA" values. If another string is used, it will be considered as not filled. FUSED means that ingress and egress spans will be fused together.
- *State*, *Country*, *Region* are not mandatory. "Region" is a holdover from the CORONET topology reference file CORONET_Global_Topology.xlsx. CORONET separates its network into geographical regions (Europe, Asia, Continental US.) This information is not used by gnpy.
- *Longitude*, *Latitude* are not mandatory. If filled they should contain numbers.
- **Booster_restriction** and **Preamp_restriction** are not mandatory. If used, they must contain one or several amplifier type_variety names separated by ' | '. This information is used to restrict types of amplifiers used in a ROADM node during autodesign. If a ROADM booster or preamp is already specified in the Eqpt sheet , the field is ignored. The field is also ignored if the node is not a ROADM node.

There MUST NOT be empty line(s) between two nodes lines

5.2 Links sheet

Links sheet must contain sixteen columns:

```
                <--           east cable from a to z                                  ␣
↪ --> <--                  west from z to                                    -->
NodeA ; NodeZ ; Distance km ; Fiber type ; Lineic att ; Con_in ; Con_out ; PMD ;␣
↪Cable Id ; Distance km ; Fiber type ; Lineic att ; Con_in ; Con_out ; PMD ; Cable Id
```

Links sheets MUST contain all links between nodes defined in Nodes sheet. Each line represents a 'bidir link' between two nodes. The two directions are represented on a single line with "east cable from a to z" fields and "west from z to a" fields. Values for 'a to z' may be different from values from 'z to a'. Since both direction of a bidir 'a-z' link are described on the same line (east and west), 'z to a' direction MUST NOT be repeated in a different line. If repeated, it will generate another parrallel bidir link between the same end nodes.

Parameters for "east cable from a to z" and "west from z to a" are detailed in 2x7 columns. If not filled, "west from z to a" is copied from "east cable from a to z".

For example, a line filled with:

```
node6 ; node3 ; 80 ; SSMF ; 0.2 ; 0.5 ; 0.5 ; 0.1 ; cableB ;  ;  ; 0.21 ; 0.2 ;  ;  ;
```

will generate a unidir fiber span from node6 to node3 with:

```
[node6 node3 80 SSMF 0.2 0.5 0.5 0.1 cableB]
```

and a fiber span from node3 to node6:

```
[node6 node3 80 SSMF 0.21 0.2 0.5 0.1 cableB] attributes.
```

- **NodeA** and **NodeZ** are Mandatory. They are the two endpoints of the link. They MUST contain a node name from the **City** names listed in Nodes sheet.
- **Distance km** is not mandatory. It is the link length.
  - If filled it MUST contain numbers. If empty it is replaced by a default "80" km value.
  - If value is below 150 km, it is considered as a single (bidirectional) fiber span.
  - If value is over 150 km the *gnpy-transmission-example*` program will automatically suppose that intermediate span description are required and will generate fiber spans elements with "_1","_2", … trailing strings which are not visible in the json output. The reason for the splitting is that current edfa usually do not support large span loss. The current assumption is that links larger than 150km will require intermediate amplification. This value will be revisited when Raman amplification is added"
- **Fiber type** is not mandatory.

  If filled it must contain types listed in eqpt_config.json in "Fiber" list "type_variety". If not filled it takes "SSMF" as default value.
- **Lineic att** is not mandatory.

  It is the lineic attenuation expressed in dB/km. If filled it must contain positive numbers. If not filled it takes "0.2" dB/km value
- *Con_in*, *Con_out* are not mandatory.

They are the connector loss in dB at ingress and egress of the fiber spans. If filled they must contain positive numbers. If not filled they take "0.5" dB default value.

- *PMD* is not mandatory and and is not used yet.

  It is the PMD value of the link in ps. If filled they must contain positive numbers. If not filled, it takes "0.1" ps value.

- *Cable Id* is not mandatory. If filled they must contain strings with the same constraint as "City" names. Its value is used to differenate links having the same end points. In this case different Id should be used. Cable Ids are not meant to be unique in general.

(in progress)

5.3 Eqpt sheet

The equipment sheet (named "Eqpt") is optional. If provided, it specifies types of boosters and preamplifiers for all ROADM degrees of all ROADM nodes, and for all ILA nodes.

This sheet contains twelve columns:

```
                  <--           east cable from a to z                  --> <--
↪west from z to a                          -->
Node A ; Node Z ; amp type ; att_in ; amp gain ; tilt ; att_out ; delta_p ; amp type ;
↪ att_in ; amp gain ; tilt ; att_out ; delta_p
```

If the sheet is present, it MUST have as many lines as there are egress directions of ROADMs defined in Links Sheet, and all ILAs.

For example, consider the following list of links (A, B and C being a ROADM and amp# ILAs):

```
A    - amp1
amp1 - amp2
Amp2 - B
A    - amp3
amp3 - C
```

then Eqpt sheet should contain:

- one line for each ILAs: amp1, amp2, amp3
- one line for each one-degree ROADM (B and C in this example)
- two lines for each two-degree ROADM (just the ROADM A)

```
A    - amp1
amp1 - amp2
Amp2 - B
A    - amp3
amp3 - C
B    - amp2
C    - amp3
```

In case you already have filled Nodes and Links sheets create_eqpt_sheet.py can be used to automatically create a template for the mandatory entries of the list.

```
$ cd $(gnpy-example-data)
$ python create_eqpt_sheet.py meshTopologyExampleV2.xls
```

This generates a text file meshTopologyExampleV2_eqt_sheet.txt whose content can be directly copied into the Eqt sheet of the excel file. The user then can fill the values in the rest of the columns.

- **Node A** is mandatory. It is the name of the node (as listed in Nodes sheet). If Node A is a 'ROADM' (Type attribute in sheet Node), its number of occurence must be equal to its degree. If Node A is an 'ILA' it should appear only once.
- **Node Z** is mandatory. It is the egress direction from the *Node A* site. Multiple Links between the same Node A and NodeZ is not supported.
- **amp type** is not mandatory. If filled it must contain types listed in eqpt_config.json in "Edfa" list "type_variety". If not filled it takes "std_medium_gain" as default value. If filled with fused, a fused element with 0.0 dB loss will be placed instead of an amplifier. This might be used to avoid booster amplifier on a ROADM direction.
- **amp_gain** is not mandatory. It is the value to be set on the amplifier (in dB). If not filled, it will be determined with design rules in the convert.py file. If filled, it must contain positive numbers.
- *att_in* and *att_out* are not mandatory and are not used yet. They are the value of the attenuator at input and output of amplifier (in dB). If filled they must contain positive numbers.
- **tilt**, in dB, is not mandatory. It is the target gain tilt over the full amplfifier bandwidth and is defined with regard to wavelength, i.e. negative tilt means lower gain for higher wavelengths (lower frequencies). If not filled, the default value is 0.
- **delta_p**, in dBm, is not mandatory. If filled it is used to set the output target power per channel at the output of the amplifier, if power_mode is True. The output power is then set to power_dbm + delta_power.

to be completed

(in progress)

5.4 Service sheet

Service sheet is optional. It lists the services for which path and feasibility must be computed with `gnpy-path-request`.

Service sheet must contain 11 columns:

```
route id ; Source ; Destination ; TRX type ; Mode ; System: spacing ; System: input␣
↪power (dBm) ; System: nb of channels ;  routing: disjoint from ; routing: path ;␣
↪routing: is loose?
```

- **route id** is mandatory. It must be unique. It is the identifier of the request. It can be an integer or a string (do not use blank or dash or coma)
- **Source** is mandatory. It is the name of the source node (as listed in Nodes sheet). Source MUST be a ROADM node. (TODO: relax this and accept trx entries)
- **Destination** is mandatory. It is the name of the destination node (as listed in Nodes sheet). Source MUST be a ROADM node. (TODO: relax this and accept trx entries)
- **TRX type** is mandatory. They are the variety type and selected mode of the transceiver to be used for the propagation simulation. These modes MUST be defined in the equipment library. The format of the mode is used as the name of the mode. (TODO: maybe add another mode id on Transceiver library ?). In particular the mode selection defines the channel baudrate to be used for the propagation simulation.
- **mode** is optional. If not specified, the program will search for the mode of the defined transponder with the highest baudrate fitting within the spacing value.

- **System: spacing** is mandatory. Spacing is the channel spacing defined in GHz difined for the feasibility propagation simulation, assuming system full load.
- **System: input power (dBm) ; System: nb of channels** are optional input defining the system parameters for the propagation simulation.
  - input power is the channel optical input power in dBm
  - nb of channels is the number of channels to be used for the simulation.
- **routing: disjoint from ; routing: path ; routing: is loose?** are optional.
  - disjoint from: identifies the requests from which this request must be disjoint. If filled it must contain request ids separated by ' | '
  - path: is the set of ROADM nodes that must be used by the path. It must contain the list of ROADM names that the path must cross. TODO : only ROADM nodes are accepted in this release. Relax this with any type of nodes. If filled it must contain ROADM ids separated by ' | '. Exact names are required.
  - is loose? 'no' value means that the list of nodes should be strictly followed, while any other value means that the constraint may be relaxed if the node is not reachable.
- **path bandwidth** is mandatory. It is the amount of capacity required between source and destination in Gbit/s. Value should be positive (non zero). It is used to compute the amount of required spectrum for the service.

CHAPTER

SIX

EXTENDING GNPY WITH VENDOR-SPECIFIC DATA

GNPy ships with an *equipment library* containing machine-readable datasheets of networking equipment. Vendors who are willing to contribute descriptions of their supported products are encouraged to submit a patch.

This chapter discusses option for modeling performance of *EDFA amplifiers*, *Raman amplifiers*, *transponders* and *ROADMs*.

6.1 EDFAs

An accurate description of the EDFA and especially its noise characteristics is required. GNPy describes this property in terms of the **Noise Figure (NF)** of an amplifier model as a function of its operating point. GNPy supports several different *noise models*, and vendors are encouraged to pick one which describes performance of their equipment most accurately.

6.1.1 Polynomial NF

This model computes the NF as a function of the difference between the optimal gain and the current gain. The NF is expressed as a third-degree polynomial:

$$f(x) = \mathrm{a}x^3 + \mathrm{b}x^2 + \mathrm{c}x + \mathrm{d}$$
$$\mathrm{NF} = f(G_{\mathrm{max}} - G)$$

This model can be also used for fixed-gain fixed-NF amplifiers. In that case, use:

$$a = b = c = 0$$
$$d = \mathrm{NF}$$

6.1.2 Polynomial OSNR (OpenROADM-style for inline amplifier)

This model is useful for amplifiers compliant to the OpenROADM specification for ILA (an in-line amplifier). The amplifier performance is evaluated via its incremental OSNR, which is a function of the input power.

$$\mathrm{OSNR}_{\mathrm{inc}}(P_{\mathrm{in}}) = \mathrm{a}P_{\mathrm{in}}^3 + \mathrm{b}P_{\mathrm{in}}^2 + \mathrm{c}P_{\mathrm{in}} + \mathrm{d}$$

6.1.3 Noise mask (OpenROADM-style for combined preamp and booster)

Unlike GNPy which simluates the preamplifier and the booster separately as two amplifiers for best accuracy, the OpenROADM specification mandates a certain performance level for a combination of these two amplifiers. For the express path, the effective noise mask comprises the preamplifier and the booster. When terminating a channel, the same effective noise mask is mandated for a combination of the preamplifier and the drop stage.

GNPy emulates this specification via two special NF models:

- The `openroadm_preamp` NF model for preamplifiers. This NF model provides all of the linear impairments to the signal, including those which are incured by the booster in a real network.
- The `openroadm_booster` NF model is a special "zero noise" faux amplifier in place of the booster.

6.1.4 Min-max NF

When the vendor prefers not to share the amplifier description in full detail, GNPy also supports describing the NF characteristics via the *minimal* and *maximal NF*. This approximates a more accurate polynomial description reasonably well for some models of a dual-coil EDFA with a VOA in between. In these amplifiers, the minimal NF is achieved when the EDFA operates at its maximal (and usually optimal, in terms of flatness) gain. The worst (maximal) NF applies when the EDFA operates at the minimal gain.

6.1.5 Dual-stage

Dual-stage amplifier combines two distinct amplifiers. Vendors which provide an accurate description of their preamp and booster stages separately can use the dual-stage model for an aggregate description of the whole amplifier.

6.1.6 Advanced Specification

The amplifier performance can be further described in terms of gain ripple, NF ripple, and the dynamic gain tilt. When provided, the amplifier characteristic is fine-tuned as a function of carrier frequency.

6.2 Raman Amplifiers

An accurate simulation of Raman amplification requires knowledge of:

- the *power* and *wavelength* of all Raman pumping lasers,
- the *direction*, whether it is co-propagating or counter-propagating,
- the Raman efficiency of the fiber,
- the fiber temperature.

Under certain scenarios it is useful to be able to run a simulation without an accurate Raman description. For these purposes, it is possible to approximate a Raman amplifier via a fixed-gain EDFA with the *polynomial NF* model using a = b = c = 0, and a desired effective d = NF. This is also useful to quickly approximate a hybrid EDFA+Raman amplifier.

6.3 Transponders

Since transponders are usually capable of operating in a variety of modes, these are described separately. A *mode* usually refers to a particular performance point that is defined by a combination of the symbol rate, modulation format, and FEC (Forward Error Correction).

The following data are required for each mode:

bit-rate Data bit rate, in Gbits $\times$ s^{-1}.

baud-rate Symbol modulation rate, in Gbaud.

required-osnr Minimal allowed OSNR for the receiver.

tx-osnr Initial OSNR at the transmitter's output.

grid-spacing Minimal grid spacing, i.e., an effective channel spectral bandwidth. In Hz.

tx-roll-off Roll-off parameter (β) of the TX pulse shaping filter. This assumes a raised-cosine filter.

rx-power-min and rx-power-max The allowed range of power at the receiver. In dBm.

cd-max Maximal allowed Chromatic Dispersion (CD). In ps/nm.

pmd-max Maximal allowed Polarization Mode Dispersion (PMD). In ps.

cd-penalty *Work-in-progress.* Describes the increase of the requires GSNR as the CD (Chromatic Dispersion) deteriorates.

dgd-penalty *Work-in-progress.* Describes the increase of the requires GSNR as the DGD (Differential Group Delay) deteriorates.

pmd-penalty *Work-in-progress.* Describes the increase of the requires GSNR as the PMD (Polarization Mode Dispersion) deteriorates.

GNPy does not directly track the FEC performance, so the type of chosen FEC is likely indicated in the *name* of the selected transponder mode alone.

6.4 ROADMs

In a ROADM (Reconfigurable Add/Drop Multiplexer), GNPy simulates the impairments of the preamplifiers and boosters of line degrees *separately*. The set of parameters for each ROADM model therefore includes:

add-drop-osnr OSNR penalty introduced by the Add and Drop stages of this ROADM type.

target-channel-out-power Per-channel target TX power towards the egress amplifier. Within GNPy, a ROADM is expected to attenuate any signal that enters the ROADM node to this level. This can be overridden on a per-link in the network topology.

pmd Polarization mode dispersion (PMD) penalty of the express path. In ps.

Provisions are in place to define the list of all allowed booster and preamplifier types. This is useful for specifying constraints on what amplifier modules fit into ROADM chassis, and when using fully disaggregated ROADM topologies as well.

CHAPTER

SEVEN

ABOUT THE PROJECT

GNPy is a sponsored project of the OOPT/PSE working group of the Telecom Infra Project.

There are weekly calls about our progress. Newcomers, users and telecom operators are especially welcome there. We encourage all interested people outside the TIP to join the project and especially to get in touch with us.

7.1 Contributing

`gnpy` is looking for additional contributors, especially those with experience planning and maintaining large-scale, real-world mesh optical networks.

To get involved, please contact Jan Kundrát or Gert Grammel.

`gnpy` contributions are currently limited to members of TIP. Membership is free and open to all.

See the Onboarding Guide for specific details on code contributions, or just upload patches to our Gerrit. Here is what we are currently working on.

7.2 Project Background

Data Centers are built upon interchangeable, highly standardized node and network architectures rather than a sum of isolated solutions. This also translates to optical networking. It leads to a push in enabling multi-vendor optical network by disaggregating HW and SW functions and focusing on interoperability. In this paradigm, the burden of responsibility for ensuring the performance of such disaggregated open optical systems falls on the operators. Consequently, operators and vendors are collaborating in defining control models that can be readily used by off-the-shelf controllers. However, node and network models are only part of the answer. To take reasonable decisions, controllers need to incorporate logic to simulate and assess optical performance. Hence, a vendor-independent optical quality estimator is required. Given its vendor-agnostic nature, such an estimator needs to be driven by a consortium of operators, system and component suppliers.

Founded in February 2016, the Telecom Infra Project (TIP) is an engineering-focused initiative which is operator driven, but features collaboration across operators, suppliers, developers, integrators, and startups with the goal of disaggregating the traditional network deployment approach. The group's ultimate goal is to help provide better connectivity for communities all over the world as more people come on-line and demand more bandwidth-intensive experiences like video, virtual reality and augmented reality.

Within TIP, the Open Optical Packet Transport (OOPT) project group is chartered with unbundling monolithic packet-optical network technologies in order to unlock innovation and support new, more flexible connectivity paradigms.

The key to unbundling is the ability to accurately plan and predict the performance of optical line systems based on an accurate simulation of optical parameters. Under that OOPT umbrella, the Physical Simulation Environment (PSE)

working group set out to disrupt the planning landscape by providing an open source simulation model which can be used freely across multiple vendor implementations.

7.3 TIP OOPT/PSE & PSE WG Charter

We believe that openly sharing ideas, specifications, and other intellectual property is the key to maximizing innovation and reducing complexity

TIP OOPT/PSE's goal is to build an end-to-end simulation environment which defines the network models of the optical device transfer functions and their parameters. This environment will provide validation of the optical performance requirements for the TIP OLS building blocks.

- The model may be approximate or complete depending on the network complexity. Each model shall be validated against the proposed network scenario.
- The environment must be able to process network models from multiple vendors, and also allow users to pick any implementation in an open source framework.
- The PSE will influence and benefit from the innovation of the DTC, API, and OLS working groups.
- The PSE represents a step along the journey towards multi-layer optimization.

7.4 License

GNPy is distributed under a standard BSD 3-Clause License.

CHAPTER

EIGHT

PHYSICAL MODEL USED IN GNPY

8.1 QoT-E including ASE noise and NLI accumulation

The operations of PSE simulative framework are based on the capability to estimate the QoT of one or more channels operating lightpaths over a given network route. For backbone transport networks, we can suppose that transceivers are operating polarization-division-multiplexed multilevel modulation formats with DSP-based coherent receivers, including equalization. For the optical links, we focus on state-of-the-art amplified and uncompensated fiber links, connecting network nodes including ROADMs, where add and drop operations on data traffic are performed. In such a transmission scenario, it is well accepted [VRS+16][BSR+12][CCB+05][ME06][SF11][JK04][DFMS04][SB11][SFP12][PBC+02][DFMS16][PCC+06][Sav05][BBS13][JA01] to assume that transmission performances are limited by the amplified spontaneous emission (ASE) noise generated by optical amplifiers and and by nonlinear propagation effects: accumulation of a Gaussian disturbance defined as nonlinear interference (NLI) and generation of phase noise. State-of-the-art DSP in commercial transceivers are typically able to compensate for most of the phase noise through carrier-phase estimator (CPE) algorithms, for modulation formats with cardinality up to 16, per polarization state [PJ01][SLEF+15][FME+16]. So, for backbone networks covering medium-to-wide geographical areas, we can suppose that propagation is limited by the accumulation of two Gaussian disturbances: the ASE noise and the NLI. Additional impairments such as filtering effects introduced by ROADMs can be considered as additional equivalent power penalties depending on the ratio between the channel bandwidth and the ROADMs filters and the number of traversed ROADMs (hops) of the route under analysis. Modeling the two major sources of impairments as Gaussian disturbances, and being the receivers *coherent*, the unique QoT parameter determining the bit error rate (BER) for the considered transmission scenario is the generalized signal-to-noise ratio (SNR) defined as

$$\mathrm{SNR} = L_F \frac{P_{\mathrm{ch}}}{P_{\mathrm{ASE}} + P_{\mathrm{NLI}}} = L_F \left(\frac{1}{\mathrm{SNR}_{\mathrm{LIN}}} + \frac{1}{\mathrm{SNR}_{\mathrm{NL}}} \right)^{-1}$$

where P_{ch} is the channel power, P_{ASE} and P_{NLI} are the power levels of the disturbances in the channel bandwidth for ASE noise and NLI, respectively. L_F is a parameter assuming values smaller or equal than one that summarizes the equivalent power penalty loss such as filtering effects. Note that for state-of-the art equipment, filtering effects can be typically neglected over routes with few hops [RNR+01][FCBS06].

To properly estimate P_{ch} and P_{ASE} the transmitted power at the beginning of the considered route must be known, and losses and amplifiers gain and noise figure, including their variation with frequency, must be characterized. So, the evaluation of $\mathrm{SNR}_{\mathrm{LIN}}$ *just* requires an accurate knowledge of equipment, which is not a trivial aspect, but it is not related to physical-model issues. For the evaluation of the NLI, several models have been proposed and validated in the technical literature [VRS+16][BSR+12][CCB+05][ME06][SF11][JK04][DFMS04][SB11][SFP12][PBC+02][DFMS16][PCC+06][Sav05][BBS13][JA01]. The decision about which model to test within the PSE activities was driven by requirements of the entire PSE framework:

i. the model must be *local*, i.e., related individually to each network element (i.e. fiber span) generating NLI, independently of preceding and subsequent elements; and ii. the related computational time must be compatible with interactive operations.

So, the choice fell on the Gaussian Noise (GN) model with incoherent accumulation of NLI over fiber spans [PBC+02]. We implemented both the exact GN-model evaluation of NLI based on a double integral (Eq. (11) of [PBC+02]) and its analytical approximation (Eq. (120-121) of [PCC+06]). We performed several validation analyses comparing results of the two implementations with split-step simulations over wide bandwidths [PCCC07], and results clearly showed that for fiber types with chromatic dispersion roughly larger than 4 ps/nm/km, the analytical approximation ensures an excellent accuracy with a computational time compatible with real-time operations.

8.2 The Gaussian Noise Model to evaluate the NLI

As previously stated, fiber propagation of multilevel modulation formats relying on the polarization-division-multiplexing generates impairments that can be summarized as a disturbance called nonlinear interference (NLI), when exploiting a DSP-based coherent receiver, as in all state-of-the-art equipment. From a practical point of view, the NLI can be modeled as an additive Gaussian random process added by each fiber span, and whose strength depends on the cube of the input power spectral density and on the fiber-span parameters.

Since the introduction in the market in 2007 of the first transponder based on such a transmission technique, the scientific community has intensively worked to define the propagation behavior of such a trasnmission technique. First, the role of in-line chromatic dispersion compensation has been investigated, deducing that besides being not essential, it is indeed detrimental for performances [CPCF09]. Then, it has been observed that the fiber propagation impairments are practically summarized by the sole NLI, being all the other phenomena compensated for by the blind equalizer implemented in the receiver DSP [CBC+09]. Once these assessments have been accepted by the community, several prestigious research groups have started to work on deriving analytical models able to estimating the NLI accumulation, and consequentially the generalized SNR that sets the BER, according to the transponder BER vs. SNR performance. Many models delivering different levels of accuracy have been developed and validated. As previously clarified, for the purposes of the PSE framework, the GN-model with incoherent accumulation of NLI over fiber spans has been selected as adequate. The reason for such a choice is first such a model being a "local" model, so related to each fiber spans, independently of the preceding and succeeding network elements. The other model characteristic driving the choice is the availability of a closed form for the model, so permitting a real-time evaluation, as required by the PSE framework. For a detailed derivation of the model, please refer to [PCC+06], while a qualitative description can be summarized as in the following. The GN-model assumes that the channel comb propagating in the fiber is well approximated by unpolarized spectrally shaped Gaussian noise. In such a scenario, supposing to rely - as in state-of-the-art equipment - on a receiver entirely compensating for linear propagation effects, propagation in the fiber only excites the four-wave mixing (FWM) process among the continuity of the tones occupying the bandwidth. Such a FWM generates an unpolarized complex Gaussian disturbance in each spectral slot that can be easily evaluated extending the FWM theory from a set of discrete tones - the standard FWM theory introduced back in the 90s by Inoue [Ino92]- to a continuity of tones, possibly spectrally shaped. Signals propagating in the fiber are not equivalent to Gaussian noise, but thanks to the absence of in-line compensation for choromatic dispersion, the become so, over short distances. So, the Gaussian noise model with incoherent accumulation of NLI has estensively proved to be a quick yet accurate and conservative tool to estimate propagation impairments of fiber propagation. Note that the GN-model has not been derived with the aim of an *exact* performance estimation, but to pursue a conservative performance prediction. So, considering these characteristics, and the fact that the NLI is always a secondary effect with respect to the ASE noise accumulation, and - most importantly - that typically linear propagation parameters (losses, gains and noise figures) are known within a variation range, a QoT estimator based on the GN model is adequate to deliver performance predictions in terms of a reasonable SNR range, rather than an exact value. As final remark, it must be clarified that the GN-model is adequate to be used when relying on a relatively narrow bandwidth up to few THz. When exceeding such a bandwidth occupation, the GN-model must be generalized introducing the interaction with the Stimulated Raman Scattering in order to give a proper estimation for all channels [CAC18]. This will be the main upgrade required within the PSE framework.

CHAPTER

NINE

API REFERENCE DOCUMENTATION

9.1 gnpy package

GNPy is an open-source, community-developed library for building route planning and optimization tools in real-world mesh optical networks. It is based on the Gaussian Noise Model.

Signal propagation is implemented in *core*. Path finding and spectrum assignment is in *topology*. Various tools and auxiliary code, including the JSON I/O handling, is in *tools*.

9.1.1 gnpy.core

Simulation of signal propagation in the DWDM network

Optical signals, as defined via `info.SpectralInformation`, enter `elements` which compute how these signals are affected as they travel through the `network`. The simulation is controlled via `parameters` and implemented mainly via `science_utils`.

9.1.1.1 gnpy.core.ansi_escapes

A random subset of ANSI terminal escape codes for colored messages

9.1.1.2 gnpy.core.exceptions

Exceptions thrown by other gnpy modules

exception gnpy.core.exceptions.**ConfigurationError**

Bases: `Exception`

User-provided configuration contains an error

exception gnpy.core.exceptions.**DisjunctionError**

Bases: *gnpy.core.exceptions.ServiceError*

Disjunction of user-provided request can not be satisfied

exception gnpy.core.exceptions.**EquipmentConfigError**

Bases: *gnpy.core.exceptions.ConfigurationError*

Incomplete or wrong configuration within the equipment library

exception gnpy.core.exceptions.**NetworkTopologyError**

Bases: *gnpy.core.exceptions.ConfigurationError*

Topology of user-provided network is wrong

exception gnpy.core.exceptions.**ParametersError**

Bases: *gnpy.core.exceptions.ConfigurationError*

Incomplete or wrong configurations within parameters json

exception gnpy.core.exceptions.**ServiceError**

Bases: Exception

Service of user-provided request is wrong

exception gnpy.core.exceptions.**SpectrumError**

Bases: Exception

Spectrum errors of the program

9.1.2 gnpy.topology

Tracking `request` for spectrum and their `spectrum_assignment`.

9.1.3 gnpy.tools

Processing of data via `json_io`. Utilities for Excel conversion in `convert` and `service_sheet`. Example code in `cli_examples` and `plots`.

CHAPTER

TEN

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[BSR+12] A. Bononi, P. Serena, N. Rossi, E. Grellier, and F. Vacondio. Modeling nonlinearity in coherent transmissions with dominant intrachannel-four-wave-mixing. *Optics Express*, 20(7):7777, 2012. URL: https://www.osapublishing.org/oe/abstract.cfm?uri=oe-20-7-7777, doi:10.1364/OE.20.007777.

[BBS13] Alberto Bononi, Ottmar Beucher, and Paolo Serena. Single- and cross-channel nonlinear interference in the Gaussian Noise model with rectangular spectra. *Optics Express*, 21(26):32254, 2013. URL: https://www.osapublishing.org/oe/abstract.cfm?uri=oe-21-26-32254, doi:10.1364/OE.21.032254.

[CAC18] Mattia Cantono, Jean Luc Auge, and Vittorio Curri. Modelling the impact of SRS on NLI generation in commercial equipment: an experimental investigation. In *Optical Fiber Communication Conference/National Fiber Optic Engineers Conference 2018*. 2018. doi:10.1364/OFC.2018.M1D.2.

[CBC+09] A. Carena, G. Bosco, V. Curri, P. Poggiolini, M. Tapia Taiba, and F. Forghieri. Statistical characterization of PM-QPSK signals after propagation in uncompensated fiber links. In *European Conference on Optical Communications, 2010*, 1–3. IEEE, 2010-09. URL: http://ieeexplore.ieee.org/document/5621509/, doi:10.1109/ECOC.2010.5621509.

[CCB+05] A. Carena, V. Curri, G. Bosco, P. Poggiolini, and F. Forghieri. Modeling of the Impact of Nonlinear Propagation Effects in Uncompensated Optical Coherent Transmission Links. *Journal of Lightwave Technology*, 30(10):1524–1539, 2012-05. URL: http://ieeexplore.ieee.org/document/6158564/, doi:10.1109/JLT.2012.2189198.

[CPCF09] V. Curri, P. Poggiolini, A. Carena, and F. Forghieri. Dispersion Compensation and Mitigation of Nonlinear Effects in 111-Gb/s WDM Coherent PM-QPSK Systems. *IEEE Photonics Technology Letters*, 20(17):1473–1475, 2008-09. URL: http://ieeexplore.ieee.org/document/4589011/, doi:10.1109/LPT.2008.927906.

[DFMS04] Ronen Dar, Meir Feder, Antonio Mecozzi, and Mark Shtaif. Properties of nonlinear noise in long, dispersion-uncompensated fiber links. *Optics Express*, 21(22):25685, 2013-11-04. URL: https://www.osapublishing.org/oe/abstract.cfm?uri=oe-21-22-25685, doi:10.1364/OE.21.025685.

[DFMS16] Ronen Dar, Meir Feder, Antonio Mecozzi, and Mark Shtaif. Accumulation of nonlinear interference noise in fiber-optic systems. *Optics Express*, 22(12):14199, 2014-06-16. URL: https://www.osapublishing.org/oe/abstract.cfm?uri=oe-22-12-14199, doi:10.1364/OE.22.014199.

[FME+16] T. Fehenberger, M. Mazur, T. A. Eriksson, M. Karlsson, and N. Hanik. Experimental analysis of correlations in the nonlinear phase noise in optical fiber systems. In *ECOC 2016; 42nd European Conference on Optical Communication*, volume, 1–3. Sept 2016. doi:.

[FCBS06] Tommaso Foggi, Giulio Colavolpe, Alberto Bononi, and Paolo Serena. Overcoming filtering penalties in flexi-grid long-haul optical systems. In *International Conference on Communications*, 5168–5173. IEEE, 2015-06. URL: http://ieeexplore.ieee.org/document/7249144/, doi:10.1109/ICC.2015.7249144.

[Ino92] K. Inoue. Four-wave mixing in an optical fiber in the zero-dispersion wavelength region. *Journal of Lightwave Technology*, 10(11):1553–1561, Nov 1992. doi:10.1109/50.184893.

[JA01] Pontus Johannisson and Erik Agrell. Modeling of Nonlinear Signal Distortion in Fiber-Optic Networks. *Journal of Lightwave Technology*, 32(23):4544–4552, 2014-12-01. URL: http://ieeexplore.ieee.org/document/6915838/, doi:10.1109/JLT.2014.2361357.

[JK04] Pontus Johannisson and Magnus Karlsson. Perturbation Analysis of Nonlinear Propagation in a Strongly Dispersive Optical Communication System. *Journal of Lightwave Technology*, 31(8):1273–1282, 2013-04. URL: http://ieeexplore.ieee.org/document/6459512/, doi:10.1109/JLT.2013.2246543.

[ME06] Antonio Mecozzi and René-Jean Essiambre. Nonlinear Shannon Limit in Pseudolinear Coherent Systems. *Journal of Lightwave Technology*, 30(12):2011–2024, 2012-06. URL: http://ieeexplore.ieee.org/document/6175093/, doi:10.1109/JLT.2012.2190582.

[PCCC07] Dario Pilori, Mattia Cantono, Andrea Carena, and Vittorio Curri. FFSS: The fast fiber simulator software. In *International Conference on Transparent Optical Networks*, 1–4. IEEE, 2017-07. URL: http://ieeexplore.ieee.org/document/8025002/, doi:10.1109/ICTON.2017.8025002.

[PCC+06] P Poggiolini, A Carena, V Curri, G Bosco, and F Forghieri. Analytical Modeling of Nonlinear Propagation in Uncompensated Optical Transmission Links. *IEEE Photonics Technology Letters*, 23(11):742–744, 2011-06. URL: http://ieeexplore.ieee.org/document/5735190/, doi:10.1109/LPT.2011.2131125.

[PBC+02] P. Poggiolini, G. Bosco, A. Carena, V. Curri, Y. Jiang, and F. Forghieri. The GN-Model of Fiber Non-Linear Propagation and its Applications. *Journal of Lightwave Technology*, 32(4):694–721, 2014-02. URL: http://ieeexplore.ieee.org/document/6685826/, doi:10.1109/JLT.2013.2295208.

[PJ01] P. Poggiolini and Y. Jiang. Recent Advances in the Modeling of the Impact of Nonlinear Fiber Propagation Effects on Uncompensated Coherent Transmission Systems. *Journal of Lightwave Technology*, 35(3):458–480, 2017-02-01. URL: http://ieeexplore.ieee.org/document/7577767/, doi:10.1109/JLT.2016.2613893.

[RNR+01] Talha Rahman, Antonio Napoli, Danish Rafique, Bernhard Spinnler, Maxim Kuschnerov, Iveth Lobato, Benoit Clouet, Marc Bohn, Chigo Okonkwo, and Huug de Waardt. On the Mitigation of Optical Filtering Penalties Originating From ROADM Cascade. *IEEE Photonics Technology Letters*, 26(2):154–157, 2014-01. URL: http://ieeexplore.ieee.org/document/6662421/, doi:10.1109/LPT.2013.2290745.

[Sav05] Seb J. Savory. Approximations for the Nonlinear Self-Channel Interference of Channels With Rectangular Spectra. *IEEE Photonics Technology Letters*, 25(10):961–964, 2013-05. URL: http://ieeexplore.ieee.org/document/6491442/, doi:10.1109/LPT.2013.2255869.

[SLEF+15] C. Schmidt-Langhorst, R. Elschner, F. Frey, R. Emmerich, and C. Schubert. Experimental analysis of nonlinear interference noise in heterogeneous flex-grid wdm transmission. In *2015 European Conference on Optical Communication (ECOC)*, volume, 1–3. Sept 2015. doi:10.1109/ECOC.2015.7341918.

[SF11] M. Secondini and E. Forestieri. Analytical Fiber-Optic Channel Model in the Presence of Cross-Phase Modulation. *IEEE Photonics Technology Letters*, 24(22):2016–2019, 2012-11. URL: http://ieeexplore.ieee.org/document/6297443/, doi:10.1109/LPT.2012.2217952.

[SFP12] Marco Secondini, Enrico Forestieri, and Giancarlo Prati. Achievable Information Rate in Nonlinear WDM Fiber-Optic Systems With Arbitrary Modulation Formats and Dispersion Maps. *Journal of Lightwave Technology*, 31(23):3839–3852, 2013-12. URL: http://ieeexplore.ieee.org/document/6655896/, doi:10.1109/JLT.2013.2288677.

[SB11] Paolo Serena and Alberto Bononi. An Alternative Approach to the Gaussian Noise Model and its System Implications. *Journal of Lightwave Technology*, 31(22):3489–3499, 2013-11. URL: http://ieeexplore.ieee.org/document/6621015/, doi:10.1109/JLT.2013.2284499.

[VRS+16] Francesco Vacondio, Olivier Rival, Christian Simonneau, Edouard Grellier, Alberto Bononi, Laurence Lorcy, Jean-Christophe Antona, and Sébastien Bigo. On nonlinear distortions of highly dispersive optical coherent systems. *Optics Express*, 20(2):1022, 2012-01-16. URL: https://www.osapublishing.org/oe/abstract.cfm?uri=oe-20-2-1022, doi:10.1364/OE.20.001022.

PYTHON MODULE INDEX

g

INDEX