Net2Plan Reference Card

José Luis Izquierdo Zaragoza (josel.izquierdo@upct.es)

Communication Network Design course, Year 2013/2014

Contents

1 Network elements

Element	Member attribute	Description
Node	id	Identifier (read-only)
	name	Node name
	position	Position in a 2D plane (x,y)
Link	id	Identifier (read-only)
	origin node	Origin node of the link (read-only)
	destination node	Destination node of the link (different than the origin node) (read-only)
	capacity	Capacity of the link (in Erlangs)
	length	Physical link length (in kilometers)
Demand	id	Identifier (read-only)
	ingress node	Source node of the demand (read-only)
	egress node	Sink node of the demand (different than the ingress node) (read-only)
	offered traffic volume	Amount of traffic offered by the demand (in Erlangs)
Route	id	Identifier (read-only)
	demand	Demand identifier (read-only)
	sequence of links	Sequence of links followed by the route
	carried traffic volume	Amount of traffic carried by the route (in Erlangs)
	backup segment list	Set of protection segments associated to the route
Protection segments	id	Identifier (read-only)
	sequence of links	Sequence of links followed by the protection segment
	reserved bandwidth	Amount of bandwidth reserved in every link in the protection segment (in Erlangs)

Table 1: Summary of network elements involved in Net2Plan, and its member attributes

2 Notation

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)

Table 2: Notation summary

3 Metrics

Type	Metric	Formula
Topology	Number of nodes	$\|N\|$
	Number of links	$\|E\|$
	Average node degree: average number of (incoming or outgoing) links per node	$\frac{\|E\|}{\|N\|}$
	Network density: it gives an idea of how dense or sparse it is the network (when no parallel links are installed, for full-mesh networks is equal to 1)	$\frac{\|E\|}{\|N\|(\|N\|-1)}$
	Network diameter: largest path among all pairs shortest paths, where $SP_{i\to j}$ represents the shortest path from node i to node j	$\max SP_{i\to j}$
	Average shortest-path length	$\overline{n} = \frac{\sum_{i\in N}\sum_{j\in N,\ j\neq i} SP_{i\to j}}{\|N\|(\|N\|-1)}$
	Average link distance	$\frac{\sum_{e\in E} l_e}{\|E\|}$
Link capacities	Total capacity installed (Erlangs)	$U_e = \sum_{e\in E} u_e$
	Average capacity installed (Erlangs)	$\frac{U_e}{\|E\|}$
	Capacity module size (Erlangs): greatest common divider among all link capacities	$\gcd e_i$
Offered traffic	Number of demands	$\|D\|$
	Average number of demands per node pair	$\frac{\|D\|}{\|N\|(\|N\|-1)}$
	Total offered traffic (Erlangs)	$H_d = \sum_{d\in D} h_d$
	Average offered traffic per demand (Erlangs)	$\frac{H_d}{\|D\|}$
Routing (carried traffic)	Link carried traffic: Equal to the sum of the carried traffic by all paths traversing the link	$y_e = \sum_{p\in P_e} x_p \quad \forall e\in E$
	Link utilization: Equal to the carried traffic by the link and the total reserved bandwidth by protection segments associated to the link, all divided by the link capacity	$\rho_e = \frac{y_e + \sum_{s\in S_e} u_s}{u_e} \quad \forall e\in E$
	Network congestion (or bottleneck utilization): Maximum load among all links	$\max_{e\in E} \rho_e$
	Throughtput (Erlangs): Total traffic injected to the network	$R_d = \sum_{d\in D} r_d$
	Total in-network carried traffic (Erlangs): Total traffic traversing the network	$Y_e = \sum_{e\in E} y_e$
	Average number of virtual hops: Total traffic traversing the network	$\overline{n}_v = \frac{Y_e}{R_d}$
	Average ingress -and egress- traffic per node (Erlangs)	$\frac{R_d}{\|N\|}$
	Average traversing traffic per node (Erlangs)	$\frac{Y_e - R_d}{\|N\|}$
	Bifurcation degree: Average number of paths carrying traffic per each demand	$\frac{\sum_{d\in D} \|\{x_p>0,\ p\in P_d\}\|}{\|D\|}$
Rate	% Lost traffic	$100 \cdot \frac{\sum_{d\in D}(h_d - r_d)}{H_d}$
	Jain's fairness index (absolute)	$\frac{\left[\sum_{d\in D}(h_d - r_d)\right]^2}{\|D\|\cdot\sum_{d\in D}(h_d - r_d)^2}$
	Jain's fairness index (proportional)	$\frac{\left[\sum_{d\in D}\frac{h_d - r_d}{h_d}\right]^2}{\|D\|\cdot\sum_{d\in D}\left(\frac{h_d - r_d}{h_d}\right)^2}$

Table 3: Notation summary

3.1 Delay: Kleinrock's independence model

In packet-switched networks, traffic sources split data into smaller pieces called **packets**, along with a header with control information. Per each received packet, switching nodes read its header and take appropriate forwarding decisions.

In real networks, traffic is highly unpredictable and often modeled as random processes. When it is said that a traffic source d generates h_d traffic units, it is referred as average traffic. As a result, link capacities would be not enough to forward traffic and nodes have to store packets in queues, so they are delayed until they can be transmitted (this delay is known as **queuing delay**). If this situation remains for a long time, queues are filled and links become **saturated**, provoking packet drops.

Network design tries to model statistically delays and drops in order to minimize their effects. In Net2Plan each link is modeled as a queue fed by a self-similar source with a given Hurst parameter, getting the whole network average delay using **Kleinrock's independence** assumption [1].

- Propagation delay per link (seconds)

$$T_e^{prop} = \frac{l_e}{v_e^{prop}} \quad \forall e \in E$$

where v_e^{prop} is the speed of light when traverses link e

- Transmission delay per link (seconds)

$$T_e^{tx} = \frac{S}{R_b u_e} \quad \forall e \in E$$

where S is the average packet length (in bits), and R_b is the capacity in bits per second per one Erlang

- Buffering delay per link (seconds)

$$T_e^{buf} = T_e^{tx} \frac{\rho_e^{[2(1-H)]^{-1}}}{(1-\rho_e)^{H/(1-H)}} \quad \forall e \in E$$

where $H \triangleq$ Hurst Parameter. Choosing $H = 0.5$ yields to same result as that predicted by M/M/1 queue model

- Delay per link (seconds)

$$T_e = T_e^{prop} + T_e^{tx} + T_e^{buf} \quad \forall e \in E$$

- Average network delay (seconds)

$$T = \frac{1}{R_d} \sum_{e \in E} y_e T_e$$

3.2 Blocking: Load-sharing model

In circuit-switched networks, traffic sources reserve a given capacity during certain time, along paths followed by traffic demands. It is possible that if a new traffic source wants to reserve resources its petition would be blocked, since it would not be enough available resources to satisfy its demand. Here, the **load-sharing model** is applied [2, 3]. According to this model, each connection request from a demand d chooses randomly a path p among the paths in P_d. The probability of choosing a path p is proportional to the traffic carried, and given by x_p/h_d. After choosing the path p, the connection is accepted if all the links in p have enough spare capacity. Otherwise, the connection is blocked. This means that there is no attempt to carry the connection in other possible routes in P_d. This is the distinguishing trait of this model. Note that thanks to this assumption, there is no overflow traffic in the network (traffic offered to a route, that if blocked, and overflows to an alternate route).

It is assumed that arrivals of **connection requests** to each link are Poisson processes, **independent link-by-link**. Therefore, blocking performance metrics can be computed using Erlang-B formula. Obviously, it has only sense when link capacities are integer.

- Blocking probability of a link

$$B_e = \text{Erlang-B}(u_e, y_e) = \frac{y_e^{u_e}/u_e!}{\sum_{i=0}^{u_e} y_e^i/i!}$$

- Average network blocking probability

$$B = \frac{1}{R_d} \sum_{e \in E} y_e B_e$$

- Worst link blocking probability

$$\max_{e \in E} B_e$$

4 References

[1] L. Kleinrock, *Queueing Systems, Volume 1: Theory*, 1st ed. Wiley-Interscience, January 1975.

[2] A. Girard, *Routing and dimensioning in circuit-switched networks.* Addison-Wesley, 1990.

[3] K. Ross, *Multiservice loss models for broadband telecommunication networks.* Springer, 1995.