# Software Development for Automatic Design and Performance Analysis of IP/MPLS Next Generation Network

**N. Kanjanasiri and C. Aswakul**
Faculty of Engineering, Chulalongkorn University, Bangkok, 10330, Thailand,
Tel: +66 (0)2 218 6908, Fax: +66 (0)2 218 6912, Email: {nat.k, chaodit.a}@chula.ac.th

***Abstract*— In this paper, an executive report has been presented on our experiences in developing a software tool for IP/MPLS NGN traffic engineering computations. The developed NGN software consists of three main modules. Firstly, the NGN design module can be used to help network engineers find the minimum link capacity needed for NGN unicast service provision (e.g. VoIP service) while maintaining the resultant QoS at a specified target level. Secondly, the NGN performance analysis module is based on a discrete-event simulation of sequential routing and trunk reservation CAC. This module can be used to help network engineers analyse the performance of NGN. It can also be used to predict the effect of link failure, traffic surge and new routing plan implementation. Finally, both the NGN design and performance analysis modules can be executed via the developed NGN GUI module. With GUI, network engineers can visualise the status of network components. Further, GUI permits both entering and editing all relevant network parameters efficiently. It is therefore very convenient for our developed NGN software to be utilised in practice and should be an indispensable traffic engineering tool for improving the NGN planning and operational tasks.**

## I. Introduction

At the rise of telecommunication needs, telecommunication markets worldwide have experienced greater transformations than ever before. The service provision paradigm in many countries has changed from the monopoly by state-own organisations into the market with many competing providers. In order to survive, each service provider must try to do their best by minimising service provision costs while maintaining the acceptable level for quality of service (QoS). Such attempts must be seriously considered throughout the whole period of network lifetime, starting from its inception planning stage towards the actual implementation as well as the network employment and maintenance stages.

Apart from the change in market structure, many new networking technologies have in the past decade been invented. Nowadays, it is clear that both legacy and contemporary telecommunication platforms are planned along various roadmaps with the same goal towards the Internet Protocol (IP) [1]. Core network services are envisioned to be all-IP and achievable IP QoS is facilitated by the traffic engineering capabilities of multi-protocol label switching (MPLS) network. This IP/MPLS suite becomes the principle part in the next generation network (NGN) [2].

This work was financially supported by the CAT Telecom PLC, Thailand, and the University-Industry Cooperation Project of Electrical Engineering Department, Faculty of Engineering, Chulalongkorn University.

From network engineering viewpoint, IP/MPLS NGN provides many technical advantages, e.g. lowering the operational cost from the economy of scales while providing a common platform flexible enough for initiation of future new services. In practice, to help engineers implement NGN, there exist many off-the-shelf softwares (e.g. [3], [4]). However, these software have been developed as a general platform that is applicable to a wide range of network technologies. Such general-purpose software is typically expensive, highly complicated and hence difficult to use by engineering practitioners.

To overcome these drawbacks, we have chosen to develop our own software specifically optimised for IP/MPLS NGN. The software goal is twofold. Firstly, the developed NGN software must be able to execute an automatic network design procedure to help engineers in network planning stage. The objective is to minimise the overall network cost to support a given traffic demand at the desired QoS level. Secondly, in the network employment stage, the NGN software must be able to identify network bottlenecks and predict the results that can be expected upon different candidate network upgrading solutions. This paper is intended as a summary report of our experiences in developing this software, which is co-designed by academic researchers at Chulalongkorn University and the practical network engineers in the industry [5].

The remainder of paper is organised as follows. In Section II, based on the framework of traffic engineering, the overall architecture of NGN software is introduced. Section III gives the mathematical formulation of automatic NGN design. Section IV explains how QoS parameters of NGN can be measured in the developed discrete-event computer simulation. Numerical examples obtained from the software are given in Section V. Finally, in Section VI, a summary of achievements relevant to this software development are given.

## II. NGN Software Architecture

Figs. 1 and 2 depict the overall architecture and data flow diagram of implemented NGN software. This software has been designed on the foundation of traffic engineering. It consists of three main modules, namely, NGN design module, NGN performance analysis module and GUI module. Network engineer can execute NGN design module in mapping from

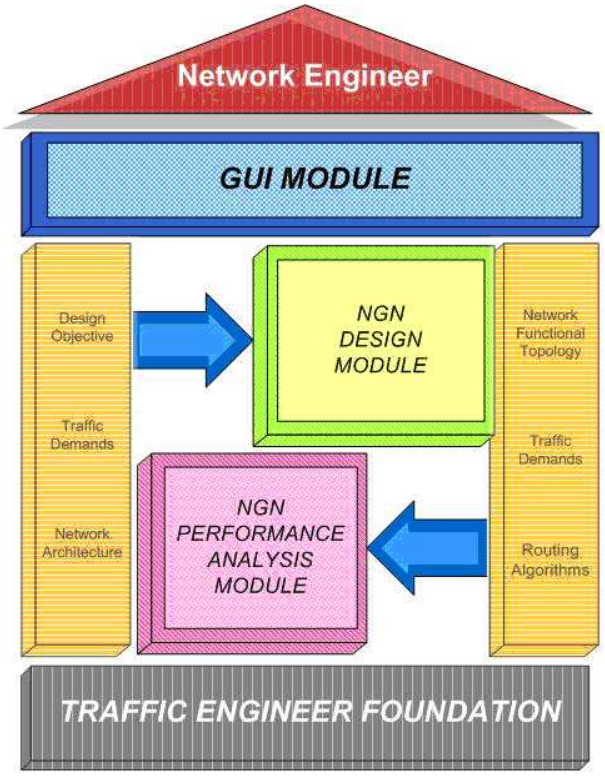


Fig. 1. Architecture of the NGN software.

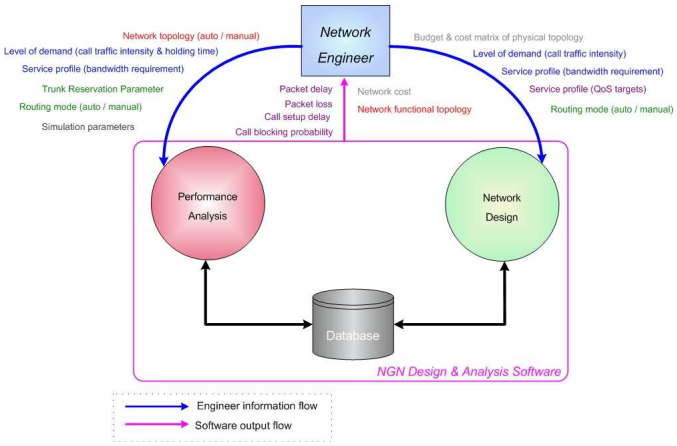


Fig. 2. Data flow diagram of the NGN software.

input design objectives, traffic demands and underlying network architecture (e.g. topology and link cost matrix) into a functional network topology with assigned link capacity matrix and a minimum-cost routing plan. Further, depending on different routing algorithms being actually employed in NGN, engineer can also use NGN performance analysis module to simulate how traffic demands would be distributed across route candidates and how such distribution affects the link utilisation as well as resultant QoS parameters. All input and output parameters can be interfaced via the GUI, which has been specifically designed in our software based on feedbacks from practical network engineers who have experimented up with our software. Therefore, GUI in our software can be easy to use and understand by those directly responsible for planning the real NGN platforms.

## III. Mathematical Formulation of NGN Design

### A. Design Objective and Assumptions

In the NGN software, the design objective is to minimise the overall network cost needed to provide a specified set of traffic demands with their desired QoS targets. These QoS targets are classified into two levels of time scale.

- Level of *connection* or *call* dynamics i.e. call blocking probability and average call setup delay.
- Level of *packet* dynamics i.e. packet loss probability and average packet delay.

Naturally different services require different levels of these QoS targets. Therefore, it is unwise to allow packet streams from different services to statically share the same portion of network resources. By doing so, it would then require that engineers must design that portion of network to satisfy the service with the most stringent target on QoS. As a result, all the other traffic streams from lower-quality service would be treated too well, which causes inefficiency in utilising network resources. For this reason, in the NGN software, we assume that each service must have its own virtual private network (VPN) and each VPN is allocated a fixed amount of capacity, which is not shared by other VPNs. This mechanism is known as *complete partitioning scheme* and has been thoroughly analysed in the past (e.g. [6]).

Another main advantage of complete partitioning scheme is that it simplifies the bandwidth dimensioning problem a great deal. In this regard, we have adopted the well-known notion of equivalent capacity function (e.g. [7]). This function defines the mapping function from the number of connections into the required link capacity to guarantee that all the connections will meet their designed QoS targets.

For notional convenience in the algorithm explanation of this paper, all demands are assumed to be of unicast type (e.g. VoIP service). Each connection is terminated between a pair of originating and destination nodes. However, the implemented NGN software can also capture other types of services equally well (e.g. multiple-unicast IP Centrex or IP PABX service, multicast VDO conference service, unicast data service and unicast signalling service). Details of automatic design and performance analysis of full service category set will be presented in the future.

### B. Automatic NGN Design Algorithm

The NGN design module first calculates the minimum cost routing plan for each origin-destination node pair by the well-known Dijkstra algorithm [8]. Here, let $r_{OD}$ denote the set of links along the minimum-cost path from origin node $O$ to destination node $D$. Also, let

$$\delta(l, r_{OD}) = \begin{cases} 1, & l \in r_{OD} \\ 0, & \text{otherwise} \end{cases}$$

and define the number of links on route $r_{OD}$ as

$$L(r_{OD}) = \sum_{l} \delta(l, r_{OD})$$

Let the demand be indexed by $s$ and its call traffic intensity between node pair $OD$ be denoted as $d_{OD}(s)$. This demand requires the call blocking probability $CBP_{OD}(s)$ not greater than target value $CBP_t(s)$; the packet loss probability $PL_{OD}(s)$ not greater than target value $PL_t(s)$; and the average packet delay $PD_{OD}(s)$ not greater than target value $PD_t(s)$. Note that the average call setup delay is herein not considered because it needs a special attention and all signalling traffics would be sent via a dedicated signalling VPN. For the design of a service VPN, the aim is to find the minimum cost $M^*(s)$ of building the VPN for demand $s$ and the capacity of each link $l$, $y_l^*(s)$ that can satisfy the QoS needs on the VPN. VPN design process for this point-to-point unicast service is explained in the following steps.

*1) Computation of the call traffic intensity offered to each link $l$:* The total demand offered to a link is obtained from a direct summation of all individual demands passing through that link. This is based on the standard assumption that each demand follows an independent Poisson process.

$$f_l(s) = \sum_{OD} d_{OD}(s)\delta(l, r_{OD}) \tag{1}$$

*2) Computation of call blocking probability:* Assume for calls between each node pair that call blocking occurs independently and identically on all the links along the minimum-cost path. The design constraint on call blocking probability on each origin-destination pair can then be written as

$$\prod_{l \in r_{OD}} [1 - CBP_l(s)] = [1 - CBP_l(s)]^{L(r_{OD})} \geq 1 - CBP_t(s)$$

where $CBP_l(s)$ is the call blocking probability of demand $s$ on link $l$. To satisfy this design constraint, we choose the appropriate value $CBP_l^*(s)$ according to the demand that passes through link $l$ along the longest path (with the number of hops equal $\max_{OD} L(r_{OD})$). It follows that

$$CBP_l^*(s) = 1 - [1 - CBP_t(s)]^{1/\max_{OD}(L(r_{OD}))} \tag{2}$$

Finally, we can calculate the call blocking probability of demand $s$ between node pair $OD$

$$CBP_{OD}(s) = 1 - \prod_{l \in r_{OD}} (1 - CBP_l^*(s)) \tag{3}$$

and the overall call blocking probability from this automatic design algorithm

$$CBP_d(s) = \frac{\sum_{OD} d_{OD}(s) CBP_{OD}(s)}{\sum_{OD} d_{OD}(s)} \tag{4}$$

*3) Computation of packet loss probability:* Similar to the call blocking computation, we assume that packet loss occurs independently and identically on all the traversing links of each minimum-cost path. It follows in the similar manner to (2) and (3) that the chosen value for packet loss probability on link $l$ for demand $s$

$$PL_l^*(s) = 1 - [1 - PL_t(s)]^{1/\max_{OD}(L(r_{OD}))} \tag{5}$$

and the packet loss probability of demand $s$ between node pair $OD$

$$PL_{OD}(s) = 1 - \prod_{l \in r_{OD}} (1 - PL_l^*(s)) \tag{6}$$

Finally, the overall packet loss probability from this automatic design algorithm can be obtained from

$$PL_d(s) = \frac{\sum_{OD} PL_{OD}(s) d_{OD}(s)(1 - CBP_{OD}(s))}{\sum_{OD} d_{OD}(s)(1 - CBP_{OD}(s))} \tag{7}$$

*4) Computation of required number of channels:* Knowing the call offered traffic $f_l(s)$ and the number of channels $z_l(s)$, we can calculate the resultant call blocking probability $CBP_l(s)$ straightforwardly from the Erlang loss function

$$\begin{aligned} CBP_l(s) &= Erlang(f_l(s), z_l(s)) \\ &= \frac{[f_l(s)]^{z_l(s)} / [z_l(s)]!}{\sum_{i=0}^{z_l(s)} [f_l(s)]^i / i!} \end{aligned}$$

In the NGN design module, we take the inverse of this Erlang loss function and the number of channels $z_l^*(s)$ required on link $l$ for demand $s$ can be computed from

$$z_l^*(s) = Erlang^{-1}(CBP_l^*(s), f_l(s)) \tag{8}$$

*5) Computation of link capacity:* In the typical case of circuit switching, knowing the number of channels is sufficient to determine the required link capacity because they are directly proportional to each other. However, in IP-based NGN, packet streams from connections sharing the same link can share the capacity of that link in a statistical manner. This is the so-called statistical multiplexing and its gain allow the economy of scales. The more connections are sharing a link, the less capacity on that link is required *per connection*. In order to capture this statistical multiplexing effect, we adopt the standard notion of equivalent capacity function [7]. That is, we can calculate the link capacity from

$$y_l^*(s) = \min\{m + \alpha'\sigma, z_l^*(s)\hat{c}\} \tag{9}$$

$$\begin{aligned} m &= R_{peak}\rho z_l^*(s) \\ \alpha' &= \sqrt{-2ln(PL_l^*(s)) - ln(2\pi)} \\ \sigma &= R_{peak}\sqrt{\rho(1-\rho)z_l^*(s)} \end{aligned}$$

$$\begin{aligned} \hat{c} = \{&\sqrt{[\alpha b(1-\rho)R_{peak} - x]^2 + 4x\alpha b\rho(1-\rho)R_{peak}} \\ &+\alpha b(1-\rho)R_{peak} - x\}/[2\alpha b(1-\rho)] \end{aligned}$$

$$\alpha = \ln(1/PL_l^*(s))$$

where $(b, \rho, R_{peak})$ is the mean burst period, source utilisation and peak bit rate of each connection. These parameters $(b, \rho, R_{peak})$ depend mainly on the type of service and their standard values are also given as defaults in the NGN software. Further, $x$ is the size of buffer (packets). In practice, $x$ must be chosen appropriately such that the average packet delay satisfies its target. In this respect, network engineers can try on vairous buffer size configurations and recursively execute the NGN design software to automatically calculate the resultant average packet delay until the average packet delay target is satisfied.

*6) Computation of average packet delay:* The average packet delay for demand $s$ between node pair $OD$ is computed from

$$PD_{OD}(s) = \sum_{l \in r_{OD}} \left[ \frac{PS_a(s) + x}{y_l^*(s)} \right] \quad (10)$$

where $PS_a(s)$ denote the average packet size of demand $s$. The average packet delay for all node pairs of demand $s$ is finally obtained from

$$PD_d(s) = \frac{\sum_{OD} PD_{OD}(s) d_{OD}(s)(1 - CBP_{OD}(s))}{\sum_{OD} d_{OD}(s)(1 - CBP_{OD}(s))} \quad (11)$$

*7) Computation of network cost:* Given $C_l$ is the cost per capacity unit of link $l$, we can compute for the cost of building VPN for demand $s$

$$M^*(s) = \sum_l y_l^*(s) C_l \quad (12)$$

and, due to the complete partitioning scheme, the total cost of building VPNs for all demands

$$M^* = \sum_s M^*(s) \quad (13)$$

## IV. Performance Analysis of NGN

The outputs from the NGN design module are the VPN topology, link capacity and resultant QoS levels. Since the network design is an iterative process, the NGN design module has been implemented with simplifications to accelerate the computational time. The main simplification is that calls between each origin-destination node pair are allowed on only one minimum-cost path. If that path is not available at the call arrival time, then that call must be blocked and lost immediately. In practice, the routing table can often provide alternatives on path selection. That is, if the path with minimum cost is not available, then the candidate paths with next minimum costs should be given a trial. For this reason, in order to capture this alternative routing, the NGN performance analysis module assumes the usage of *sequential routing* [9], whereby paths are tried in the sequence according to their increasingly sorted costs.

Furthermore, in the NGN design module, a call is allowed to use any portion of link capacity for its access. This is called *complete sharing policy* of call admission control (CAC). However, it is noted that engineers may want to execute CAC in a more controllable way. For instance, demands may be granted access to network resources in their respective order of priority. Therefore, in the NGN performance analysis module, we have adopted the well-known *trunk reservation policy* for CAC [10], [11]. That is, a call will be accepted into a link only when the remaining capacity on that link after accepting the call remains greater than a threshold called *trunk reservation parameter*.

Both sequential routing and trunk reservation features have been integrated into the discrete-event simulation (e.g. [12]) in the NGN performance analysis module. Fig. 3 gives an overall summary of this NGN simulation. To execute this simulation, due to the complete partitioning of VPNs, engineers can analyse each of the VPNs separately. The NGN software has been implemented by using Java running on the Red hat LINUX operating system. It is then very convenient for engineers to execute our software in simultaneous processes, each is responsible for the performance analysis of individual VPNs.

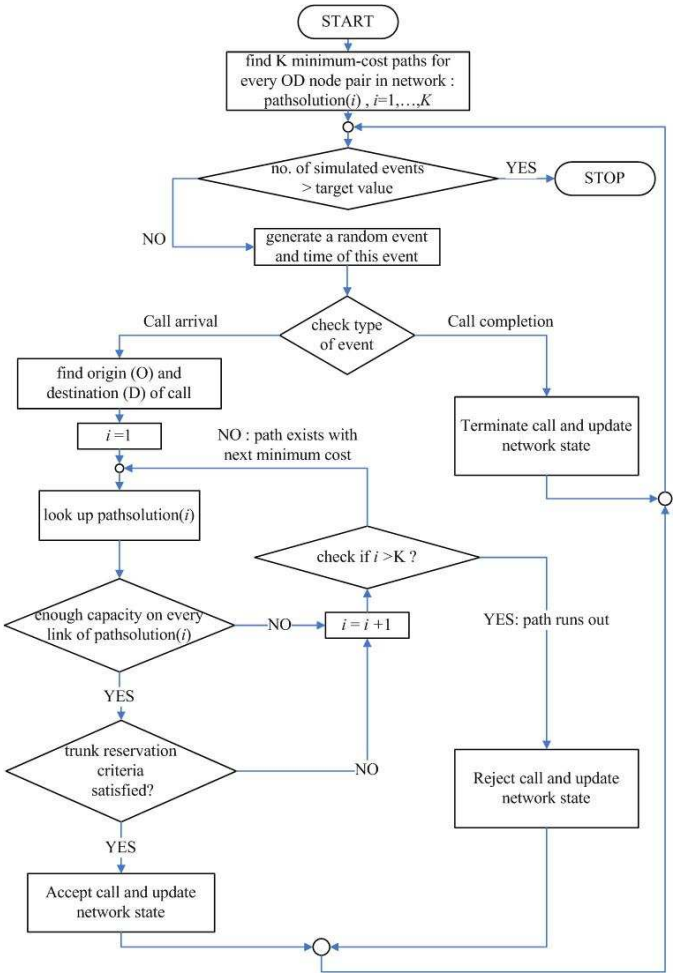


Fig. 3. Computer simulation flowchart of NGN.

## V. Numerical Examples

To demonstrate the NGN software, we have chosen a particular network design and analysis scenario, based on a modified version of a company's core network which spans across the whole country of Thailand. Fig. 4 depicts the topology and link cost (per capacity unit) of the chosen network example. The traffic intensity is set to 100 call Erlangs for every node pair. The call service is the standard IP-based VDO phone with equivalent capacity function parameters $b$ = 0.12s, $\rho$ = 0.25, $R_{peak}$ = 0.768 Mbps, $x$ =15 kBytes, $PS_a$ = 1.5 kBytes and average call holding time = 30 mins. The QoS targets are set to $CBP_t$ = 0.3, $PL_t$ =0.001 and $PD_t$ = 0.4s. Fig. 5 presents the result of link capacity computation from the NGN design module and Fig. 6 guarantees that the design target in terms of call blocking probability has been well met.

From the functional topology in Fig. 5, we further analysed the performance of network by invoking the NGN performance

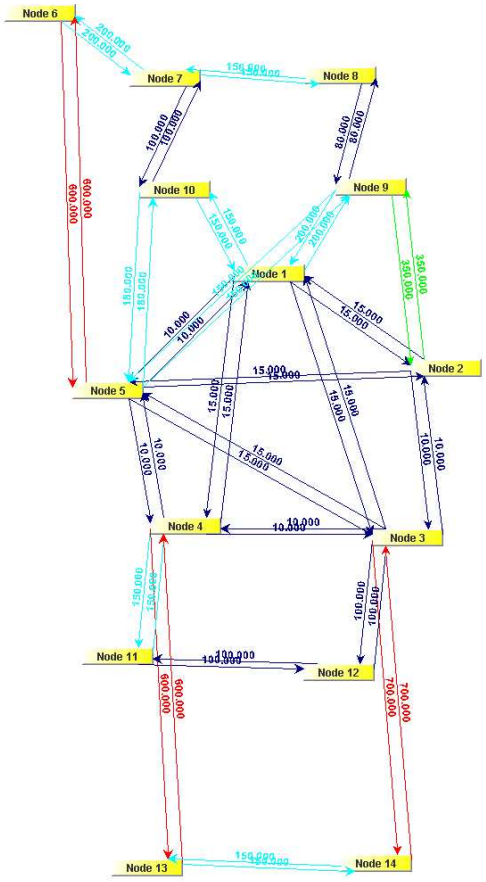


Fig. 4. Topology and link cost of example network.

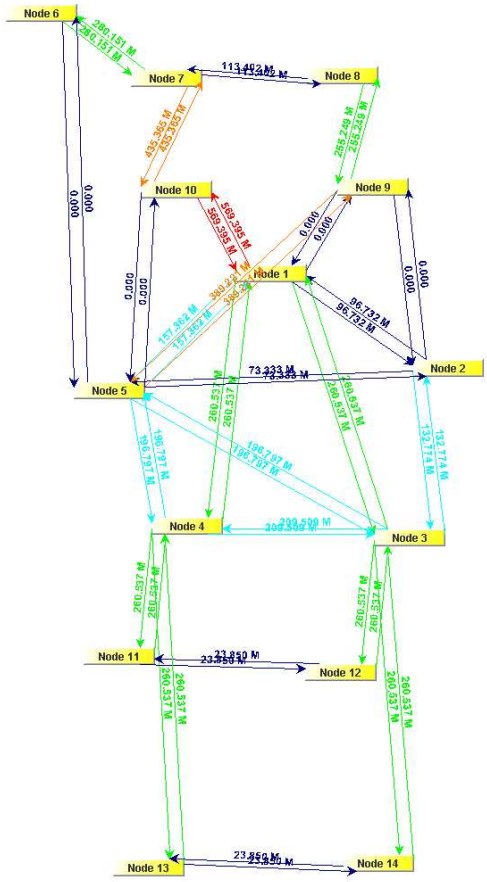


Fig. 5. Link capacity assignment from NGN design module.

analysis module. The call blocking probability from the NGN performance analysis module is depicted in Fig. 7. Here, we set the maximum number of candidate routes to 5 and the trunk reservation to 80 percent of link capacity. Note that all the bottleneck links with high level of call blocking probability can be easily visualised by our GUI.

It should also be noted that other performance and design parameters can also be visualised conveniently via our developed GUI. And apart from this network example, we have also tried out experiments on various situations. For instance, we can adopt the NGN software to study the effect of link failure, traffic surge, rerouting strategy, upgraded link capacity and predicting the network bottlenecks. It is therefore believed that the NGN software herein developed can be a useful tool for traffic engineering tasks of network engineers responsible for the well-beings of NGN.

## VI. Conclusion

This paper has summarised our experiences in developing the NGN design and analysis software, which is the result of collaborative efforts from academic researchers at Chulalongkorn University and the practical network engineers in the industry. This software has been designed on the foundation of traffic engineering. It consists of three main modules, namely, NGN design module, NGN performance analysis module and GUI module. In the NGN design module, the design objective is to minimise the overall network cost needed to provide a specified set of traffic demands with their desired QoS targets. An automatic NGN design algorithm has been given for the unicast service provision, based on the complete partitioning scheme of VPNs. In the NGN performance analysis module, the objective is to analyse the QoS parameters in both the time scale of call and packet dynamics (e.g. call blocking probability, packet loss probability, average packet delay). This module relies on the technique of discrete-event computer simulation, where the standard sequential routing and trunk reservation CAC have been emulated. Preliminary results from both the NGN design and performance analysis modules have been given to demonstrate the software capability in dealing with the real-world network size. Due to the involved complexities, it is recommended that the platform for running the NGN software should be workstation class. In our project, we have simultaneously utilised two workstation machines, each with dual 2.8 GHz Xeon CPU, 2 GB RAM. Finally, all input and output parameters can be interfaced via the developed GUI, which has been specifically designed in our software based on feedbacks from practical network engineers who have experimented up with our software. Therefore, GUI in our software can be easy to use and understand by those directly responsible for planning the real NGN platforms.

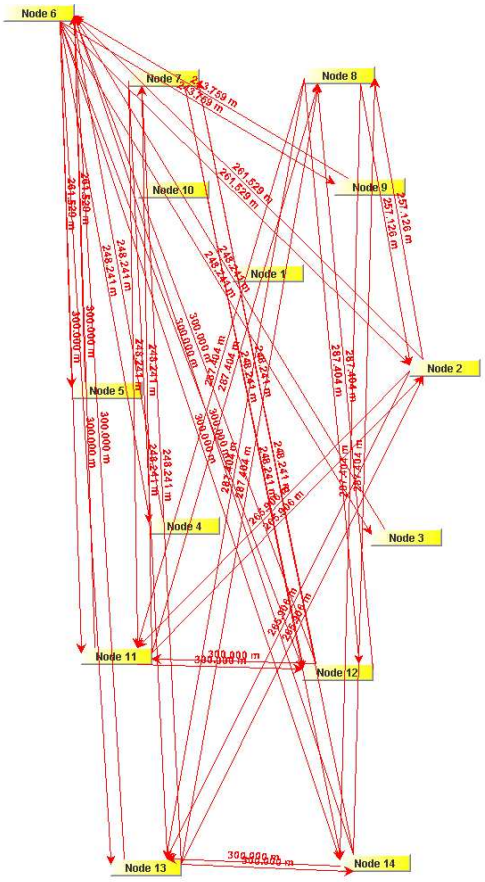


Fig. 6. Call blocking probability between each OD from NGN design module.

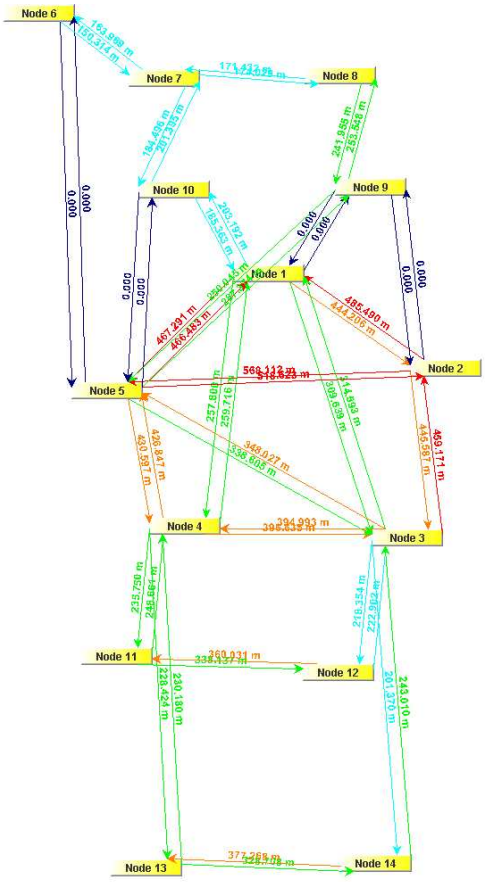


Fig. 7. Call blocking probability on each link from NGN performance analysis module.

## ACKNOWLEDGEMENT

The authors would like to acknowledge the practical suggestions throughout the period of NGN project from Dr Anotai S., Dr Anak M. and engineers from CAT Telecom PLC, Thailand. The authors would also like to acknowledge all the efforts of every member of $(NG)^2$ (Next Generation Network Group) at Chulalongkorn University, Thailand.

## REFERENCES

[1] ITU-T Recommendation Y. 2261, *PSTN/ISDN Evoluation to NGN*, 2006.
[2] ITU, *A Handbook on Internet Protocol (IP)-Based Networks and Related Topics and Issues*, 2005.
[3] www.opnet.com
[4] www.vpisystems.com
[5] C. Aswakul, et. al., *Development of Computer Simulation Software for Design and Analysis of Next Generation Telecommunication Network*, Final Report Submitted to the Research and Development Department, CAT Telecom PLC, Thailand, 2007.
[6] K. W. Ross, *Multiservice Loss Models for Broadband Telecommunication Networks*, Springer Verlag, 1995.
[7] R. Guerin, H. Ahmadi and M. Naghshineh, "Equivalent Capacity and Its Application to Bandwidth Allocation in High-Speed Networks," *IEEE Journal on Selected Areas in Communication*, Vol. 9, No. 7, pp. 969–981.
[8] D. Bertsekas and R. Gallager, *Data Networks*, 2nd Ed., Prentice Hall, 1992.
[9] A. Girard, *Routing and Dimensioning in Circuit-Switched Networks*, Addison-Wesley, 1990.
[10] J. W. Roberts, "Teletraffic Models for the Telecom I Integrated Services Network," Proceedings of 10th International Teletraffic Conference, Montreal, Canada, 1983, paper 1.1-2.
[11] C. Aswakul and J. Barria, "Analysis of dynamic service separation with trunk reservation policy," *IEE Proceedings - Communications*, Vol. 149, No. 1, February 2002, pp. 23-28.
[12] A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*, 3rd Ed., McGraw-Hill, 2000.