Towards Network X-ities From a Topological Point of View: Evolvability and Scalability

Mung Chiang and Michael Yang
Electrical Engineering Department, Princeton University, NJ 08544

Abstract

Network are designed not just to maximize efficiency of performance metrics, but also to enhance various X-ities: evolvability, scalability, reliability, verifiability, deployability, adaptability... As a step towards a rigorous understanding of network X-ities, this paper presents some analytic frameworks and quantitative methodologies on evolvability and scalability. The focus of this investigation in this broad and under-explored area is on the topological features of networks. We present an Evolvable Network Design (END) Tool using dynamic programming methods to design the multi-phase deployment of a network so that early phase designs are most evolvable to later phases. Then we use several quantitative measures to compare the scalability properties of star, ring, mesh, and access-aggregator-core topologies. The purpose of this paper is to explore the important but fuzzy notions of X-ities in the relatively under-articulated area of network robustness, using simple quantitative formulations that suffice to illustrate some of the key issues at stake.

1 Introduction

1.1 Beyond performance efficiency and towards X-ities

Communication networks are not designed only for efficiency of *performance* metrics in terms of throughput, latency and distortion, but also for *robustness* in terms of important network X-ities: evolvability, scalability, reliability, verifiability, manageability, deployability, survivability, adaptability ... Compared with the standard performance metrics, most of these X-ities are much less well-understood, often without any theoretical foundations, quantitative frameworks, or even units of measurement. Yet network X-ities are crucially important if we are to analyze existing networks and design future ones properly.

These X-ities can be viewed as different manifestations of 'robustness' of a network [4], robustness to link or node failures, to malicious attacks, to design flaws, to future evolutions in the temporal dimension, to the need for growth in the spatial dimension, and to unreliable network elements and users. On most of the X-ities, very few quantitative prior results have been obtained by the academic research community. On the other hand, network system engineers at service providers are more likely to fully appreciate the importance of X-ities in designing, deploying, and operating networks, although they almost always have to resort to engineering intuitions and ad hoc heuristics today. We need to bridge this significant gap between practitioners and theoreticians, and try to turn network design principles based on X-ities from religious slogans to quantitative methodologies.

Proper characterization of these X-ities will also lead to new insights on important networking principles: the pros and cons of layering architecture (which is perhaps designed to enhance evolvability, scalability and manageability, although almost all the recent cross layer papers discuss layering only from the performance perspective), the pros and cons of overlay networking (which seems to be another general principle to make networks evolvable and scalable), the end-to-end principle, the state-less argument, the need to duplicate certain functions in protocols at different layers, the division between core and access networks, and the tradeoff between local computation and global communication.

1.2 Overview

In general, we would like to quantify and then leverage the notion of flexible network design, where flexibility may mean that given a small ϵ change in the external environment (user requirements, available technology...), only a small δ change in the network architecture is needed. Specifically, we will focus on the following evolvability problem encountered in almost every network deployment. Because capital expenditures are constrained and customer demands only grow gradually, networks are deployed in multiple phases. The engineering design challenge is: how to deploy today's network that will be easily evolvable to meet tomorrow's uncertain demands? Currently, there exists no systematic methodology to design a phase 1 deployment that is optimal in the sense of being most readily upgrade-able into phase 2 to satisfy new customer demands without disturbing existing connections.

People often agree that the Internet architecture is 'scalable'. If we are given two networks with certain functions to be running over the networks, can we tell which is more scalable and how much more scalable? We also know that when the size of a system keeps increasing pass a certain threshold, its structure and form may also have to change qualitatively. Does scalability mean that the structures and forms, and the underlying architectural principles, of a network can remain the same no matter how large the network becomes?

As a step towards a rigorous understanding of network X-ities, this paper presents some analytic frameworks and quantitative methodologies on evolvability and scalability, focusing on the topological (rather than protocol) aspects of network architectures.

1.3 Access-aggregator-core architecture

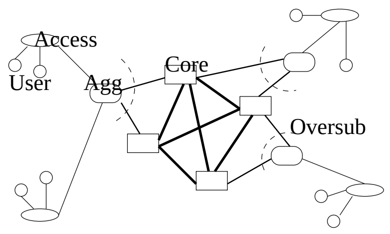

Figure 1: Parts of a 3-stage access-aggregator-core-network.

Many of the quantitative methodologies in this paper will be illustrated through examples of access-aggregator-core (AAC) networks, which are often used in practical deployment of telecommunication networks by service providers in the U.S., especially for metropolitan area networks. While the Internet is certainly not being deployed by a single carrier following any particular pre-determined topology, it shares some of the basic features of aggregations and core connections in AAC networks.

AAC networks can be parameterized by natural numbers $l \geq 2$, which denotes the number of stages in the network. Figure 1 shows parts of a 3-stage network. The basic idea is to use inexpensive and small access switches to connect a number of end hosts nearby, then aggregate the traffic from access switches to more expensive aggregator switches, allowing oversubscription at the aggregator so as to leverage the statistical multiplexing possibility during traffic aggregation, and finally connect each aggregator switch to two high-end core switches. This 'dual-homing' from aggregators to the core is used to ensure there exists more than one path connecting an end host to the core. The core is full meshed to ensure carrier class quality, *i.e.*, each core switch is connected to every other core switch. Then the topology is mirrored from the core to the other end hosts through the aggregator and access switches. Thus there are in fact $2l - 1$ stages in what we call a l-stage network.

2 Network Evolvability

2.1 Motivation and related work

Evolvability is an important engineering principle that has been discussed extensively in fields like neural nets and computer systems. Evolution of biological and social networks that grow according to natural mechanisms, like preferential attachment, have also been quantitatively studied in the last few years, *e.g.*, surveyed in [5]. However, for communication networks that grow under specific technological and economic constraints, evolvability remains a highly fuzzy concept, even though its important has been recognized for a long time for both telecom networks and the Internet (*e.g.*, [1, 2]). Roughly speaking, 'evolvability' refers to the ability of a network to reconfigure its architecture and protocols as user demands, technology cost-effectiveness, and socio-economic trends vary over time. To initiate a rigorous and quantitative understanding of this fuzzy notion of evolvability, we propose to formulate the following specific problem that shows up in almost every practical deployment of networks by service providers.

Suppose a service provider would like to deploy a new network to cover bandwidth demands from users located in a region. It is expected that there will be a large number of users after 5 years, but the service provider does not wish to deploy the optimal network with respect to the year 5 customer demand all in year 1. Due to both budget constraints on capital expenditure in each year and gradual growth in customer demands over the years, the entire network must be deployed in phases. Each phase in general should not contain just the bare minimum set of switches and links to cater the existing customer demands in that phase. That would represent the greedy solution which may restrict the evolution path of the network into the future.

For example, a single aggregator switch might be adequate to cover all the demands in phase 1. But if indeed only one aggregator is deployed and all geographically diverse users are connected to this switch, then when the number of users in each subregion becomes big enough to warrant a dedicated switch, the users connected to the far away switch deployed in phase 1 are traversing an unnecessarily long and expensive link. By looking ahead into future phases, it may be more cost-effective to deploy multiple switches in phase 1, if the resulting maintenance and inflation costs are more than compensated for by a more cost-effective network topology at the end of the last phase. The question now becomes, under budgetary constraints, how much overbuilding of the network should be allowed in each phase, beyond covering the existing demands and in anticipation of growth in the number of users and demands per user in future phases?

In a realistic model, there will be many more expense-revenue tradeoff considerations, constant parameters, and optimization variables than in the above illustrative example. But the essence of network evolvability in multi-phase deployment remains the same: how to strike the best balance between being greedy and allowing overbuilding in each phase?

2.2 END Tool

Our Evolvable Network Design (END) Tool applies the dynamic programming principle to solve the *sequential decision problem* of multiple phase network deployment. A dynamic program can be visualized by a trellis diagram [3] of nodes and edges. Each column of the trellis is indexed by k: phases of network deployment. Each node in the trellis represents a physical layout state, comprising of customers, switches and links, and each edge represents a possible transition between states. Incremental network deployments update a state at phase k to another one at phase $k+1$. There is an edge cost between two states, which is the sum of the costs required to build and maintain infrastructure minus the customer profits gained at time k. If it is impossible to reach a state from another one, the cost is infinite.

The objective of evolvable network design is to obtain the most profitable network deployment throughout a fixed number of phases, based on market forecasts on customer locations and bandwidth demands [1]. This automated design process is conducted off-line before network deployment, and incorporates a variety of topological and technical constraints. We summarize only the basic notation and model below [2]. Some of the concrete numerical examples are for AAC topologies, but the applicability of **END Tool** certainly spans beyond AAC networks.

The state data structure consists of the following basic fields: the location coordinates of the access, aggregator, and core switches at phase k, and location coordinate and bandwidth requirement from each user, the pair of switches connected by each link (including customer-access, access-aggregator, aggregator-core, and core-core links). The control data structure consists of the following basic fields: new switches and new links. Together with the new customers data structures, the control data structure updates the state data structure by conducting the appropriate constructions.

In addition to budget constraints per phase, the control spaces are constrained in many ways. For example, each type of switch can only be built within a certain subset of the points on the grid. In order for a link to be built between switches, the switches at the end points must have already been deployed and there must be vacant ports on the switches. The number of ports on each switch is one of the many parameters that can be specified, each with a different maximum data rate it can support. All topological constraints of AAC networks must be satisfied, *e.g.*, dual-homing from aggregator to core and full mesh within the core. Furthermore, each customer must be connected to an access switch within a radius of r units with sufficient bandwidth support. We use a recursive function to determine if the network elements are connected according to the rules.

The trellis diagram must be established before running the standard recursive dynamic programming algorithm. There are two ways to build the trellis. One is to start backward from a known final state where a greedy algorithm finds the network topology based on the customer demands at that final phase, and possible states in earlier phases are constructed from phase $N-1$ to phase 1. Alternatively, one can construct all the possible states at phase $k+1$ from the states at phase k and all the allowed control strategies. The first approach would generate substantially fewer states but is not as general as the second approach. The two approaches represent different tradeoffs between computational tractability and dynamic programming optimality. In both cases, due to budget constraints, only a limited segment of the network can be constructed beyond meeting the requirements from existing customers. Also notice that some dangling segment of the network may be constructed without serving any customer during a given phase, if at a later phase, by constructing a few links connecting that segment to the core,

[1]These are sometimes based on accurate customer polling, and sometime rather inaccurate, in which case stochastic dynamic programming or multiple runs of **END Tool** are carried out based on different scenarios.

[2]Complete modeling and implementation details can be obtained in our **END Tool** User Manual [6].

a substantial population of customers can be served.

Costs of edges in the trellis diagram also need to be assigned. Each cost consists of two parts: expense and revenue. Expense include both one-time building expense (incurred only at the phase of building the switch or link) and recurrent operating expense (incurred in all the phases after the construction) for each type of switches and each type of links. Revenue consists of both a one-time customer signup revenue and a recurrent customer payment revenue.

In general, being greedy in a phase runs the risk of cornering future phases into highly suboptimal design space, since once an end-to-end connection is established, it cannot be torn down as long as there are active traffic through it. The optimal evolution path strikes the right balance between building ahead of time in anticipation of customer growth and avoiding overbuilding to reduce unnecessary costs. Once the trellis diagram is constructed, applying the standard dynamic programming recursion on the trellis is straight-forward. Since the state-spaces are deterministic and finite, finding optimal network evolution over an evolution of N phases is equivalent to computing the minimum cost path across the trellis diagram.

2.3 Numerical examples

We have conducted extensive tests on **END Tool** through large scale numerical simulations and developed several heuristics to reduce the computational load. The first set of examples is on a large network deployment scenarios over 12 phases, taking into account a variety of engineering tradeoffs. Here we assume that there is a fixed final state (FFS) that is deemed as the optimal at the last phase. This assumption substantially reduces the state space sizes along the trellis diagram. Figures 2, 3, and 4 show the phase 1, 4, and 12 (final phase) network topology along the optimal evolution path as computed by **END Tool**. End users, access switches, aggregator switches, and core switches are shown in different colors. Figure 5 shows one of the suboptimal evolution paths at phase 4. Its suboptimality is not apparent at all at this phase, but are manifested in later phases. The total deployment cost is -8012 units for the optimal evolution path, but can be as high as -140 units for the suboptimal ones. (A negative deployment cost means that net profit can be generated even during network deployment.)

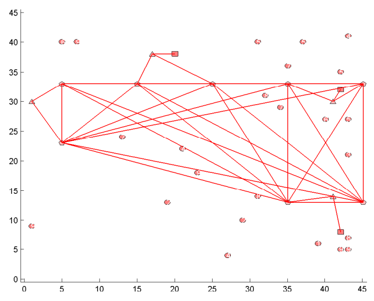

Figure 2: FFS example: optimal evolution phase 1.

The second set of examples is on a medium size network deployment scenario over 10 phases, where we can generate a much larger state space and still finish computing the optimal evolution path on a desktop PC in minutes. Here the final state is not pre-determined (non-fixed-final-state NFFS), but becomes a by-product of the construction of the trellis diagram from phase 1 to phase 10. Figures 6, 7, and 8 show the network topology at phases 1, 3, and 10 in the optimal evolution path as computed by **END Tool**. Figure 9 shows the final phase network topology in one of the suboptimal evolution paths. Obviously, compared to Figure 8, no one would have

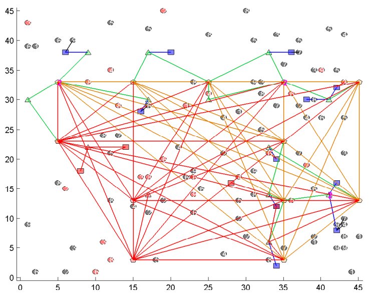

Figure 3: FFS example: optimal evolution phase 4.

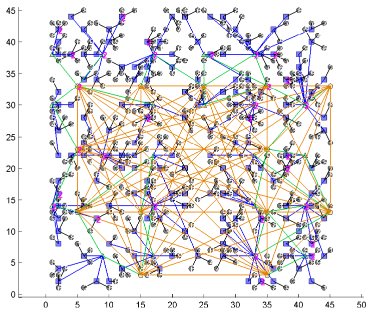

Figure 4: FFS example: optimal evolution final phase.

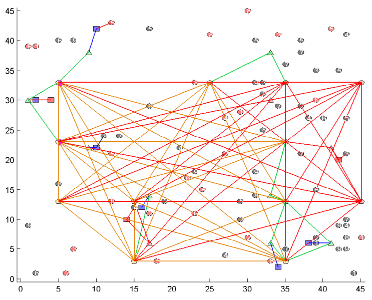

Figure 5: FFS: An example of suboptimal evolution phase 4.

designed the network as in Figure 9 had the entire deployment been carried out in one single phase. The **END Tool** confirm and quantify the intuition that striking the appropriate balance between being greedy in each phase and anticipating the future demands dramatically enhance network efficiency in the end phase. The deployment cost for the optimal evolution path is 3797 units whereas that for the suboptimal ones can be as high as 28634 units.

3 Network Scalability

3.1 Motivation and related work

The 'scalability argument' has often been used for the Internet and wireless ad hoc networks, but remains a fuzzy notion in many respects. Scalability can be discussed as a property of a

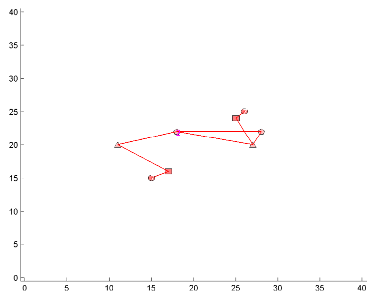

Figure 6: NFFS example: optimal evolution phase 1.

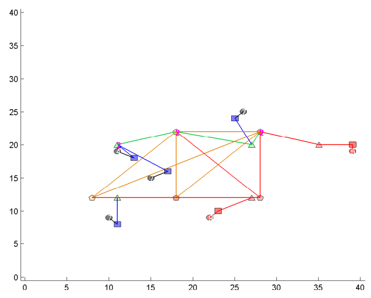

Figure 7: NFFS example: optimal evolution phase 3.

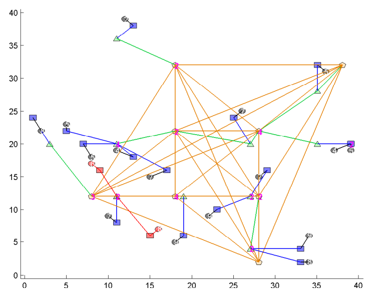

Figure 8: NFFS example: optimal evolution final phase.

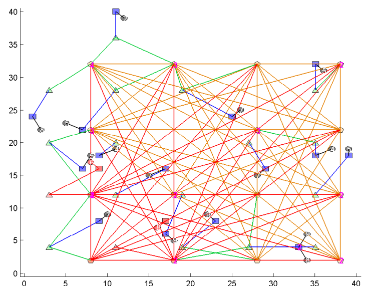

Figure 9: NFFS: An example of suboptimal evolution final phase.

network architecture or a control protocol. Sometimes, it refers to the hard limit of how large a network can be, as a function of the architecture or protocol. If the network can grow without bound, then we are interested in how does a QoS metric like throughput, or the overhead in a protocol like a routing protocol, grows with the network size. If the QoS metric grows quickly, *e.g.*, superlinearly, or the protocol overhead grows slowly, *e.g.*, sublinearly, it is usually viewed as scalable.

Similar to other X-ities studies in this paper, we will focus on scalability of *topological features* of network architecture, and characterize the tradeoff between 'limits on size' and 'QoS metrics' to quantify the notion of scalability. We are also interested in answering the question: "Will an increase in the *size* of a network force a change in the *form* of the network?" We will then use these quantitative frameworks to compare the scalability property of several common network topologies: star topology, ring topology, full mesh topology, and AAC topology.

3.2 Scalability metrics

We use 'network elements' as a generic phrase to refer to different kinds of switches. The size of a network element refers to the number of ports on the switch that can be supported by the switch fabric and backplane.

Definition 1 Maximum size N_{max} *of a network is the maximum number of users that can be supported, as a function of both the network topology and the sizes of the network elements.*

Sometimes, a network architecture can be naturally expanded to support more users. For example, the AAC architecture can be expanded from 3-stage to 4-stage or even larger number of stages. In these cases where the network topologies form a natural ordering, we would like to quantify how scalable a network topology of a particular order l is.

Definition 2 Scalability index ρ_l *is the ratio of the maximum size* $N_{max,l}$ *of a* l*th order network and the maximum size of a* $(l-1)$*th order network (i.e., the smallest number of users that would make* l*th order network necessary when a lower order network is no longer big enough):*

$$\rho_l = \frac{N_{max,l}}{N_{max,l-1}}.$$

A misconception that needs to be avoided is that scalability can be characterized by the maximum size alone. Some network architectures may be able to support a large number of users, almost unconstrained by the size of the network elements, but QoS may deteriorate rapidly as the number of users increases. A scalable architecture should be one that can support a large number of users without sacrificing QoS too much. This tradeoff can be quantified in general as a *size-QoS product*: the product of maximum network size and some measure of QoS.

For example, many QoS metrics, like reliability and latency, are monotonic functions of the average number of hops $H(N)$ it takes to reach one user from another when there are N users. We will use $H(N)$ as the QoS metric in the examples in this section. We define the following quantity $\Gamma(N)$, essentially as the reciprocal of the size-QoS product:

Definition 3 $\Gamma(N)$ *is the product of* $H(N)$ *and* $M(N)$*, where* $M(N)$ *is the size of the bottleneck network element needed to support* N *users.*

Notice that a smaller $\Gamma(N)$ implies a more scalable network.

3.3 Comparison of common topologies

We use the following notation for AAC architecture. Let N be the total number of end users, v be the number of end users that each access switch port can support, κ_l be the oversubscription factor for a network with l stages, N_l be the number of switches at the lth stage, and M_l be the number of ports per switch at the lth stage. The oversubscription factor is larger in networks with a larger number of stages, because there are more links aggregating into the last aggregator stage, allowing a more aggressive statistical multiplexing and higher κ_l.

Because the core network is required to be full mesh, there is a limit on how many end users can be supported by an AAC network with a fixed number of stages. We have the following

Proposition 1 *The maximum size of l-stage AAC networks is:*

$$N_{max,l} = \frac{v\kappa_l}{8}(M_l + 1)^2,$$

and assuming the same port count in the core switches for AAC networks with different numbers of stages, the scalability index is

$$\rho_l = \frac{\kappa_l}{\kappa_{l-1}}.$$

This result quantifies how scalable AAC network architecture is and allows quantitative trade-off analysis between scalability (which increases as oversubscription κ_l increases) and reliability (which decreases as oversubscription increases). It also clarifies where the scalability comes from as the number of stages increases. The benefit to scalability in AAC networks comes from the ability to allow more aggressive oversubscription as the number of stages increases. This proposition also quantifies the notion that as the size of a network increases, a change in the form becomes necessary at a critical size that depends on the technology of network elements (in this case, the port count in the core switch).

We now use the notion of size-QoS product to compare AAC topology with three other common topologies: star, full mesh, and ring. Specifically, we use $\Gamma(N)$, the product of $H(N)$ and $M(N)$, to quantitatively confirm the rule-of-thumb in network design that AAC strikes a good balance between scalability and performance.

Proposition 2 $\Gamma(N)$ *of AAC networks is is $\mathcal{O}(\sqrt{N})$. For star networks, ring networks, and full mesh networks, $\Gamma(N)$ is $\mathcal{O}(N)$.*

A numerical example illustrating the precise formula of $\Gamma(N)$ is shown in Figure 10. The AAC network used here is a typical 3-stage one with an oversubscription factor of 5. The graph confirms the intuition that ring networks are more scalable than star or mesh networks, but also indicates that AAC networks' scalability property surpasses that of ring networks even for medium size networks.

4 Concluding Remarks

Aimed at a rigorous understanding of network X-ities, we present the **END Tool** to design the multi-phase deployment of a network so that early phase designs are evolvable to later phases. Then we use several metrics to quantify the higher scalability of AAC topology compared with star, mesh, and ring topologies. This paper is only a first step away from the performance-centric mind-set and towards network X-ities. The exploration is limited in several respects. Many other important X-ities are not discussed, such as verifiability, manageability, deployability... We focus on the network topology aspects, which certainly do not exhaust the possible formulations of X-ities. And the mathematical tools utilized in this paper are relatively simple. More sophisticated machineries need to be developed to study X-ities more comprehensively.

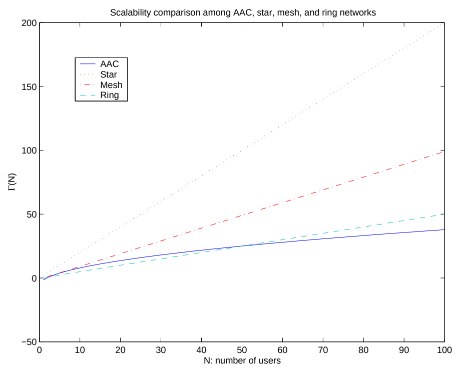

Figure 10: Numerical example of scalability comparison among AAC, star, full mesh, and ring networks by the $\Gamma(N)$ metric.

These limitations further highlight the difficulties associated with characterizing, understanding, and designing network X-ities. The practical and theoretical importance of this emerging research area, however, is by no means diminished by these technical difficulties and intellectual challenges. This paper illustrates that it is both possible and fruitful to develop quantitative methodologies for network X-ities.

We are not advocating to abandon performance metrics. Rather, in networking problems where network X-ities can be quantified, a rigorous *tradeoff* analysis can be conducted between X-ities metrics and performance metrics. Networks should be designed to be on the *Pareto optimal surface* above the dimensions representing both types of metrics.

Acknowledgement

We would like to thank stimulating discussions on network robustness and X-ities with Venkat Anantharam, Vincent Chan, John Doyle, Sandy Fraser, P. R. Kumar, Steven Low, Sekhar Tatikonda, John Tsitsiklis, and long-term collaboration with SBC.

References

[1] J. F. Bars and B. Loyer, “Reliability and evolvability: requirements for the design of ISDN communicatoin software,” *IEEE J. Sel. Area Comm.,* vol. 6, no. 8, pp. 1405-1413, Oct. 1988.

[2] R. K. Berman, S. F. Knapp, and R. F. Baruzzi, “Evolvability of the advanced intelligent network,” *Proc. IEEE Inter. Conf. Comm.,* 1991.

[3] D. Bertsekas, *Dynamic Programming and Optimal Control,* Athena Scientific, 2001.

[4] J. M. Carlson and J. C. Doyle, “Highly optimized telerance: robsustness and design in complex systems,” *Phy. Rev. E,* vol. 84, pp. 2529-2532, 2000.

[5] S. N. Dorogovtsev and J. F. F. Mendes, *Evolution of Networks,* Oxford University Press, 2002.

[6] M. Yang and M. Chiang, *END Tool User Manual,* Princeton University, 2004.