GENESIS

1 Introduction

GENESIS is a computational agent based network formation model for the Internet at the Autonomous System (AS) level. GENESIS models each AS as an individual agent which asynchronously engages in peering strategy, provider and peer selection. GENESIS is modular and extensible. It can incorporate different peering strategies, provider and peer selection criteria, traffic matrix and geographic presence characteristics, cost and revenue parameters etc. Inter-AS routing in GENESIS follows the shortest path "valley-free" routing policy. GENESIS can thus be used by researchers to experiment with

We consider a population $\mathcal{N}$ of N Autonomous Systems, or just "nodes". Each node is present in a given set of locations which correspond to IXPs. An internetwork traffic matrix gives the average traffic volume sent from each network to every other network. The model only captures the interconnections between distinct networks and does not consider intradomain activity and connectivity. No particular business type is associated with any network and they may act as transit providers, content sources, access providers, enterprise networks or any combination of these functions, depending on their generated/consumed traffic, transit pricing, or geographical presence. The geographic locations, transit prices and traffic of each network remain constant during the timescales of interest in this model. Instead, we focus on the impact of set of peering strategies on the dynamics of the provider and peer selection process: how does the internetwork and the economic fitness of networks change with time as new peering strategies are introduced in the strategy pool?

2 Model Description

2.1 Locations and Geographical presence

Each node is placed in one or more *locations*. We may think of a "location" as a distinct Internet Exchange Point, for instance. The *world* consists of G_M locations. Node x is located at a given set $G(x)$ of locations. $G(x)$ is an input to this model. Two nodes *overlap* if they are co-located in at least one location. For node x, $O(x)$ denotes the set of nodes that overlap with $O(x)$. Two nodes x and y must overlap if they are to establish a link between themselves. An exception to this rule exists if x and y are non-overlapping $Tier-1$ nodes. Non-overlapping $Tier-1$ nodes can form a peering link among themselves. This exception is introduced to ensure that the network always remains connected.

2.2 Traffic matrix and transit traffic

Each element T_{xy} of an N-by-N interdomain traffic matrix T denotes the average traffic volume generated by node x and consumed by node y. GENESIS does not capture connectivity and traffic flow within an AS. Therefore, $Txx = 0$. If aggregate across all consumers of traffic, we have that x generates traffic $V_G(x)$.

$$V_G(x) = \sum_{i=1}^{N} T_{xi} \tag{1}$$

If we aggregate across all producers of traffic, we have that x consumes traffic $V_C(x)$.

$$V_C(x) = \sum_{i=1}^{N} T_{ix} \tag{2}$$

The *transit* traffic volume $V_T(x)$ is the traffic volume that is neither generated nor consumed by x - it only passes through x enroute to its destination. If $V_O(x)$ denotes total *outbound* traffic at x then $V_T(x)$ is given by:

$$V_T(x) = V_O(x) - V_G(x) - V_C(x) \tag{3}$$

GENESIS only captures timescales during which interdomain traffic matrix remains constant. Thus, $V_G(x)$ and $V_C(x)$ remain constant during model execution. However, transit traffic for a node depends on underlying network topology and therefore may change if the network topology changes. The total traffic volume $V(x)$ of node x is the aggregate of its local traffic ($V_G(x)$ and $V_C(x)$) and its transit traffic.

$$V(x) = V_T(x) + V_G(x) + V_C(x) \tag{4}$$

2.3 Economic attributes

The economic attributes of each node include its transit prices, transit cost and revenue, peering costs and $fitness$.

Prices: Each node x maintains a vector of prices $P(x)$. Each element of the vector corresponds to the price the node has in each region in which it is present. The price of node x in each region is determined randomly independent of its prices in other regions and the prices of co-located nodes. The random prices are chosen from a specific range.

Transit Cost: Let x be a *transit customer* of y. Let $P_y(x)$ be the lowest transit price of y across all regions in which x and y overlap. If $V_P(x)$ is the traffic exchanged between x and y over their transit link then the transit payment from x to y is given by:

$$TC(x) = P_y(x) \times V_P(x)^{\tau} \tag{5}$$

where τ is the transit traffic exponent used to capture *economies of scale*. The transit revenue of y is the aggregate of transit costs incurred by all transit customers of y. Let $C(y)$ denote the set of all transit customers of y. Then its transit revenue is given by:

$$TR(y) = \sum_{\forall x \in C(x)} TC(x) \tag{6}$$

Peering Cost: Nodes engaging in settlement free peering relationship share underlying peering costs. GENESIS models both public and private peering between nodes. Private peering is a one-to-one relationship while public peering is a one-to-many relationship. Private peering is modeled as a direct link between the peers whereas public peering is modeled as peering over common peering infrastructure at an IXP. For simplicity we assume that a network aggregates all its public peering traffic at a single port.

Private Peering Cost: Let $V_{PP}(x, y)$ be the traffic exchanged between network x and its peer y over a private peering link. The cost of private peering for x is given by:

$$PC_{prv}(x) = \alpha \times \sum_{y} V_{PP}(x, y)^{\beta} \tag{7}$$

β is the peering traffic exponent used to capture economies of scale in peering costs.

Public Peering Cost: Let $V_{PP}(x, z)$ be the traffic exchanged between network x and its peer z over public peering infrastructure. The cost of public peering for x is given by:

$$PC_{pub}(x) = \alpha \times (\sum_{z} V_{PP}(x, z))^{\beta} \tag{8}$$

The total peering cost of x is given by:

$$PC_{total}(x) = PC_{prv}(x) + PC_{pub}(x) \tag{9}$$

Fitness: The fitness of a node x represents its net profit and is given by:

$$F(x) = TR(x) - TC(x) - PC_{pub}(x) - PC_{prv}(x) \tag{10}$$

Stubs (nodes without any customers) have $TR(x) = 0$. Thus, stubs have negative fitness. *Providers* (nodes with at least one transit customer) may or may not have positive fitness depending on their costs and revenue.

2.4 Peering

Each node x has a peering strategy $S(x)$. Two nodes x and y are potential peers if:

1. x and y overlap
2. x and y do not have a customer-provider relationship
3. the traffic exchanged between x and y over the proposed peering link, $V_{PP}(x, y)$, will be greater than a certain threshold Ω

The above conditions define *universal* peering constraints for all pairs of nodes. However, potential peers x and y can establish peering relationships iff they satisfy the constraints of each other's peering strategies:

1. x satisfies the constraints of $S(y)$
2. y satisfies the constraints of $S(x)$

Thus, peering is a bilateral decision process unlike provider selection (explained later). Depeering, however, is a unilateral decision by each peer. If the traffic exchanged between x and y over the peering link is less than a certain threshold Ψ then they peer privately, otherwise they peer publicly. $V_{PP}(x, y)$ includes traffic generated and consumed by both networks i.e. $T_{xy} + T_{yx}$ and also any traffic that would be exchanged between nodes in the customer trees of x and y routed through that peering link.

The set of all public and private peers of x is denoted by $PP_{pub}(x)$ and $PP_{prv}(x)$ respectively. $PP(x)$ denotes the set of all peers of x.

$$PP(x) = PP_{pub}(x) \cup PP_{prv}(x) \tag{11}$$

Let $y \in S(x)$ denote that y satisfies the constraints of peering strategy of x. Algorithm 1 shows the peering acquisition algorithm for node x. Algorithm 2 shows depeering decision by node x.

Algorithm 1 Peer acquisition

Construct $PP_{ptl}(x) = \{y : y \in O(x),\ y \notin C(x),\ x \notin C(y),\ y \in S(x),\ V_{PP}(x,y) > \Omega\}$

For $\forall y \in PP_{ptl}(x)$

If $x \in S(y)$

If $T_{xy}^{PP} \geq \Psi$

Do $PP_{prv}(x) = PP_{prv}(x) + y$

$PP_{prv}(y) = PP_{prv}(y) + x$

Else

Do $PP_{pub}(x) = PP_{pub}(x) + y$

$PP_{pub}(y) = PP_{pub}(y) + x$

End

Do $PP(x) = PP_{prv}(x) \cup PP_{pub}(x)$

Algorithm 2 Depeering

For $\forall y \in PP(x)$

If $y \notin S(x)$

If $y \in PP_{prv}(x)$

Do $PP_{prv}(x) = PP_{prv}(x) - y$

Else

Do $PP_{pub}(x) = PP_{pub}(x) - y$

End

Do $PP(x) = PP_{prv}(x) \cup PP_{pub}(x)$

2.5 Provider Selection

A node requires a *provider* if it cannot reach all other nodes in the network through its peers and customers. Node x selects node y as provider if:

1. y overlaps with x
2. y is not a peer of x
3. y is not a customer of peer of x
4. y is "larger" than x for appropriate metrics of node size
5. y is the least expensive among all nodes that satisfy the previous constraints.

GENESIS relates node size with the geographic footprint of the node as well as the volume of transit that it carries. Thus, for y to be a provider of x, y should be present in at least as many number of locations as x and should carry transit traffic strictly larger than x. Formally, the provider of node x denoted by $R(x)$ is given by:

$$R(x) = \arg \min_{y \in O(x)} P(y) : V_T(y) > V_T(x), |G(y)| \geq |G(x)|, y \notin PP(y), y \notin C(PP(x)) \tag{12}$$

Ties between nodes with the same price are broken deterministically, based on unique node identifier. If a node cannot find any provider that can fulfill the above criteria then the node does not choose any provider and acquires $Tier-1$ status. All $Tier-1$ nodes are connected to one another through peering links even if they do not overlap in order to ensure that the network remains connected.

Algorithm 3 shows the decision process for provider selection for node x.

Algorithm 3 Provider Selection

Construct $O'(x) = \{y : y \in O(x), |G(y)| \geq |G(x)|, V_T(y) > V_T(x)\}$

If $O'(x) = \emptyset$

do $R(x) = \emptyset$

$PP(x) = PP(x) + z \quad : R(z) = \emptyset, \forall z \in \aleph$

Else

do $R(x) = q : q \in O'(x), P_x(q) < P_x(m) \, \forall m \in O'(x)$

2.6 Routing

Interdomain routing in GENESIS follows the shortest path subject to two common policy constraints: "prefer customer over peer over provider links" and satisfy the "valley-free" routing property. GENESIS only allows this particular routing policy.

2.7 Network formation process

An execution or sample path of GENESIS proceeds in discrete time units called *iterations*. Each network *plays* asynchronously during each iteration once. Thus, an iteration encapsulates N discrete sequential actions each by a unique node. The order in which nodes play during an iteration is determined at the beginning of the sample path and remains the same throughout the execution. Each node carries out the following actions while playing:

1. Examine depeering any existing peers which do not conform to the constraints of current peering strategy
2. Examine peering with all potential peers which conform to the current peering strategy (and which "bilaterally" accept peering request under their own peering strategies)
3. Provider selection
4. Peering strategy update

The fitness of each node is re-computed at the end of each iteration. The state of the network at any point in time can be defined based on the connections and peering strategy of all nodes. Two states **A** and **B** are distinct if they differ in terms of the underlying network topology or the peering strategy of one or more nodes. Even if we start with the same population of nodes and the same initial topology, two sample paths can result in two distinct equilibria as a result of different playing orders.

If none of the nodes has adjusted its connectivity and peering strategy in a particular iteration i.e. the state of the network has not changed during an iteration as compared to the previous iteration, then can be shown that there will be no changes in subsequent iterations, and we say that GENESIS has reached an *equilibrium*. The network formation process in GENESIS is deterministic (the playing order does not

change during a sample path), and so it can have one of two possible outcomes: either convergence to an equilibrium or a limit cycle in which the network moves repeatedly through the same sequence of states. An equilibrium results when none of the nodes changes its connections or peering strategy during an iteration. A limit cycle, or *oscillation*, on the other hand, results when the state of the network $S(t_k)$ in the k'th iteration is identical to the state $S(t_{k-m})$ in the $(k-m)$'th iteration with $m > 1$. The length of the limit cycle in that case is m. Note that there is no possibility for chaotic trajectories because the system has a finite number of states, and so if it does not converge to an equilibrium, it will eventually return to a previously visited state. We denote the state of an individual node x at time t by $s(x,t)$.

Algorithm 4 Network formation process

Input:

$X \leftarrow$Ordered set of nodes $\{x_0, x_1, x_2, ..., x_N\}$

Begin $t \leftarrow 0$

Start Iteration $i \leftarrow 0$

Play node x_i

Examine depeering for node x_i

If x_i depeered any peer in examination above

Do *Update* $V_T(x_k)$ $\forall x_k \in X$

Examine peering for node x_i

If x_i acquired any new peer in examination above

Do *Update* $V_T(x_k)$ $\forall x_k \in X$

Examine provider selection for node x_i

If x_i changed its provider in examination above

Do *Update* $V_T(x_k)$ $\forall x_k \in X$

If $i == N$

Do *Update* $F(x_k)$ $\forall x_k \in X$

Do *Record* $S(t)$

If $t > 0$ && $S(t) == S(t-1)$

End *Declare Equilibrium*

Else If $t > 1$ && $S(t) == S(t') : \exists! \, t' < t-1$

End *Declare Oscillation*

Else

$t \leftarrow t+1$

Go to $\rightarrow$**Start Iteration**

Else

$i \leftarrow i+1$

Go to $\rightarrow$**Play**

2.7.1 Proof: If $s(x,t) = s(x,t-1)\ \forall x \in N$ the network is at equilibrium

Let $s(x,t-1)$ and $s(x,t)$ be the state of node x at the end of iteration of $t-1$ and t respectively. Assume $s(x,t) = s(x,t-1)\ \forall x \in N$. This implies that no node changed its connectivity and peering strategy during iteration t. Let x be the first node to play in iteration $t+1$.

Let x acquire a new peer y in iteration $t+1$. But this contradicts the conditions for peering action which requires that for a change to happen in $PP(x)$ there should be a change in transit volume or peering strategy of x or some other node. The actions of x during iteration t (depeering, peering, provider selection, peering strategy update) result in any change iff there is a change in peering strategy and/or transit volume of x or some other node during the interval between $x's$ actions in iteration t and $t+1$. Since there was no change in iteration t, $PP(x)$ cannot change in iteration $t+1$. The same logic can be applied to depeering, provider selection and peering strategy update. This implies that actions of x will not result in any change in iteration $t+1$. Since x did not cause any change and there was no change to the network in iteration t, the node playing after x will also not cause any change so on until all nodes have played in iteration $t+1$. The process will repeat till infinity without any change to the network. The network is therefore at equilibrium.

2.7.2 Proof: If an network reappears in the same state without convergence to equilibrium it will oscillate till infinity

A network shows oscillatory behavior if $S(t) = S(t+T)\,,\ T > 1,\ t > 1$ and $S(t) \neq S(t+t')\ \forall t' < T$. We show that if a network appears in the same state in two non-consecutive iterations its state will show an oscillatory pattern with time period T till infinity. Let (i,t) denote the time at which ith node plays during iteration t. The actions of ith player at time (i,t) are a response to the state of the network at time $(i-1,t)$. Since the actions of a node are deterministic, its response to a particular initial state will always be the same.

Let a network reappear in state S' after an interval T as shown in figure 1. For simplicity we assume that S' coincides with an iteration boundary. Let t' be the first iteration to end with the appearance of the state S'. Let x be the first node to play during an iteration followed by y. Since the order in which nodes play during an iteration does not change x and y will play at the same positions during all iterations. Let $A(x,t,i)$ denote the actions of x during iteration t when x is the ith node to play during the iteration. The actions $A(x,t'+1,1)$ of x are a response to the state of the network S' resulting in network state $S(t'+1,1)$.

$$S(t'+1,1) = S' + A(x,t'+1,1)$$

x is followed by y.

$$S(t'+1,2) = S(t'+1,1) + A(y,t'+1,2)$$

Let $t'+T$ be the first time when S' reappears. When x plays the resulting state is:

$$S(t'+1+T,1) = S(t'+T) + A(x,t'+1+T,1)$$

But $S(t'+T) = S'$

$$\implies S(t'+1+T,1) = S' + A(x,t'+1+T,1)$$

since actions of x are deterministic its response to S' will be the same as before.

$$\implies A(x,t'+1+T,1) = A(x,t'+1,1)$$

$$\implies S(t'+1+T,1) = S' + A(x,t'+1,1)$$

$$\implies S(t'+1+T,1) = S(t'+1,1)$$

As a result, the response of y will also repeat.

$$\implies S(t'+1+T,2) = S(t'+1,2)$$

The responses of all nodes will repeat in the same fashion until we see S' again at time $t'+2T$. The process will repeat in an oscillatory fashion till infinity.

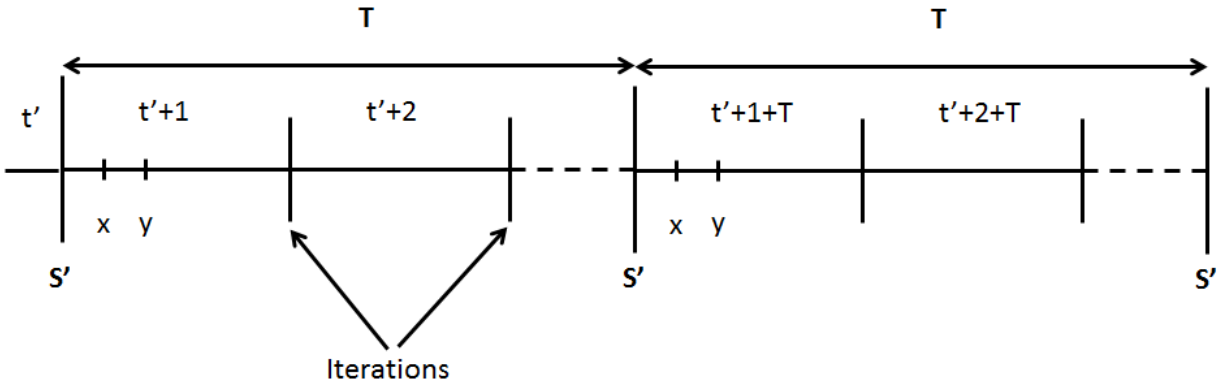

Figure 1: Reappearance of internetwork state

2.8 Peering Strategies

The following peering strategies have been incorporated in GENESIS.

2.8.1 Open

A node that uses this strategy agrees to peer with any other node that it overlaps with (except direct customers).

2.8.2 Selective

A node x that uses this strategy agrees to peer with node y if $\frac{V_x}{V_y} > \sigma\,(\sigma > 0)$. In practice, there is a wide range of Selective peering strategies with several additional constraints and parameters (e.g., a minimum link capacity or a minimum number of pointsof- presence). Selective strategy in GENESIS is only a model that aims to capture the essence of those requirements through a simple formula and single parameter .

2.8.3 Restrictive

A node that uses this strategy does not peer with any other node unless if that is mandatory to maintain global reachability ("peering-by-necessity").

2.8.4 Cost Benefit Analysis

This peering strategy allows a node x to accurately determine the effect of peering with node y on its fitness. If the fitness of x improves as a result of peering with y then x retains y as peer otherwise depeers it. Algorithms 1 and 2 show the decision process for node x for peering and depeering using cost benefit analysis, respectively.

Algorithm 5 Peering using Cost Benefit Analysis

Do $F'(x) = F(x)$

$PP(x) = PP(x) + y$

$Compute\, V_T(x)$

$Compute\, F(x)$

If $F(x) < F'(x)$

Do $PP(x) = PP(x) - y$

Algorithm 6 Peering using Cost Benefit Analysis

Do $F'(x) = F(x)$

$PP(x) = PP(x) - y$

$Compute\, V_T(x)$

$Compute\, F(x)$

If $F(x) < F'(x)$

Do $PP(x) = PP(x) + y$

2.8.5 Paid peering

A node x using this strategy charges all its peers y a peering fee. The peering fee is set to half the transit fee that y would have to pay x, if y were its customer i.e. $\frac{P_x(y)}{2}$. Peering costs still apply to both nodes. If both peers, x and y follow this strategy then x pays y the peering fee if $P_x(y) < P_y(x)$. The peering fee per unit of traffic charged by y in this case is $P_y(x) - P_x(y)$.

3 Default model

This section describes a certain instance of GENESIS that we refer to as the "Default model". Table 1 shows the values of the parameters for the default model along with brief explanation.

Table 1: Input Parameters

Parameter, Symbol, Description	**Value**	**Explanation**
Number of ASes N	500	Simulation time constraints
Number of geographic locations G_{Max}	50	Based on approximate ratio of IXPs to ASes in the Internet [1]
Geographic expanse distribution	Zipf(1.6)	Based on data about number of participants at each IXP collected from PeeringDB [1]. $G(x)$ assigned randomly to each node
Maximum expanse for an AS	15	
Generated traffic distribution	Zipf(1.2)	Produces a heavy-tailed distribution of outgoing traffic. With this distribution, 0.1% of the ASes generate nearly 28% of the total traffic. This is consistent with the behavior reported in [2], [3] & [4], which show that the traffic produced by high-ranking ISPs and content providers follows a Zipf distribution. $V_G(x)$ assigned randomly to each node
Consumed traffic distribution	Zipf(0.8)	Produces heavy-tailed distribution of incoming traffic, similar to measured traffic distribution at Georgia Tech. $V_C(x) \propto \|G(x)\|$, rationale being that a node with large expanse will also have a large number of access customers
Mean consumed traffic	500 Mbps	
Private peering threshold Ψ	50 Mbps	[5]
Transit cost multiplier range $P(x)$	\$[35,45]/Mbps per iteration	Parameterized based on IP transit prices reported by VoxNet [6]. $P(x)$ assigned randomly
Transit cost exponent τ	0.75	Parameterized based on data from [7] and [5]
Peering cost multiplier α	\$20/Mbps per iteration	
Peering cost exponent β	0.40	
Selective peering ratio σ	2.0	

3.1 Geographic distribution

The *world* in the Default model consists of 50 locations. The maximum number of locations in which a node can be present is 15. Algorithm 7 is used to assign geographic locations $G(x)$ to each node x.

Algorithm 7 Location assignment to nodes

Input: Zipf distribution parameter: $g_z = 1.6$, $max(|G(x)|) = 15$, $N = 500$, $G_M = 50$

1 Populate set F_G with $max(|G(x)|)$ elements with fractions in the interval (0,1) based on zipf distribution with shape parameter g_z

Comment: Fractions sum to 1. The above step effectively creates $max(|G(x)|)$ bins numbered from 1 to $max(|G(x)|)$.

2 Sort F_G

3 $F_G(j) = round(F_G(j) * N)$ $1 \leq j \leq max(|G(x)|)$

Comment: The above step assigns to each bin j the number of nodes which will have expanse j

4 Create $X = \{x_i : 1 \leq i \leq N\}$ random ordering of N nodes

5 $j \leftarrow 1$

6 $i \leftarrow 1$

7 **If** $F_G(j) == 0$

Do $j \leftarrow j + 1$

8 $|(G_{x_i})| = j$

9 $i \leftarrow i + 1$

10 **If** $i == N$

Go to Step 13

11 $F_G(j) \leftarrow F_G(j) - 1$

12 Go to step 7

13 Randomly assign $|G(x_i)|$ locations to each node x_i

14 End

3.2 Traffic matrix calculation

Algorithm 8 Traffic matrix calculation

Input $N = 500$, Zipf distribution parameters for generated and consumed traffic: t_z^G and t_z^C, Maximum traffic consumed by a node: $max(V_C)$

1 Sort set of nodes $\aleph$ in ascending order of geographic expanse $|G(x)|$

2 Generate fractions $frac_C$ in the interval (0,1) based on zipf distribution with shape parameter $= t_z^C$

Comment: Total 500 fractions generated. The fractions sum to 1. The above list of fractions give the fraction of total traffic consumed by an individual node

3 Sort $frac_C$ in ascending order

4 Based on the two sorted lists above (Consumed traffic fractions and networks sorted by geographic expanse), assign fractions to networks $frac_C(x_i)$. *Comment:* 1-to-1 mapping. The lowest fraction goes to the network with smallest expanse and the highest fraction goes to the network with largest expanse. Ties are broken by node id

5 Determine total traffic in the network based on $max(V_c)$

$$V_{Total} = \lceil \frac{max(V_C)}{max(frac_C(x))} \rceil$$

6 Generate fractions $frac_G$ in the interval (0,1) based on zipf distribution with shape parameter $= t_z^G$

Comment: Total 500 fractions generated. The fractions sum to 1

7 Randomly assign fractions to each node: $frac_G(x)$

8 Calculate generated traffic

$$V_G(x) = round\,(frac_G(x) \times V_{Total})$$

9 Update generated traffic

$Residue = V_{Total} - \sum_{x=1}^{N} V_G(x)$

If $Residue > 0$

$V_G(y) = V_G(y) + Residue \ : \ V_G(x) = max(V_G(x)) \ \ \forall x \in \aleph$

10 Compute traffic matrix T

Set $V_C(x) = 0 \ \forall x \in \aleph$

Repeat the following steps $\forall x \in \aleph$

1. $T_{xy} = round\,(V_G(x) \times frac_C(y)) \ \forall y \in \aleph,\ y \neq x, \quad V_C(y) = V_C(y) + T_{xy}$
2. $Residue = V_G(x) - \sum_{y=1, y \neq x}^{N} T_{xy}$
3. If $Residue > 0$

   $T_{xy} = T_{xy} + Residue \ : \ V_C(y) = min(V_C(z)) \, \forall z \in N,\ z \neq x, \ V_C(y) = V_C(y) + T_{xy}$

3.3 Initial topology creation

Network formation process in GENESIS begins with an initial network. The following algorithm is used to construct initial topology.

Algorithm 9 Initial topology construction

1 Randomly assign $R(x) = y$: $|G(y)| > |G(x)|$, y is not in the customer tree of x

2 If $R(x) == \emptyset$

Make x a T1 node and peer x with all T1 nodes

4 References

References

[1] "PeeringDB." `http://www.peeringdb.com`, September 2010.

[2] H. Chang, S. Jamin, Z. Mao, and W. Willinger, "An Empirical Approach to Modeling Inter-AS Traffic Matrices," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, 2005.

[3] A. Feldmann, N. Kammenhuber, O. Maennel, B. Maggs, R. De Prisco, and R. Sundaram, "A Methodology for Estimating Interdomain Web Traffic Demand," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, 2004.

[4] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian, "Internet Inter-domain Traffic," in *Proceedings of ACM SIGCOMM*, 2010.

[5] "Peering Strategy Survey." `http://drpeering.net/white-papers/Peering-Policies/A-Study-of-28-Peering-Policies.html`.

[6] "VoxNet IP Services." `http://www.voxel.net/ip-services`.

[7] H. Chang, S. Jamin, and W. Willinger, "To Peer or Not to Peer: Modeling the Evolution of the Internet's AS-level Topology," in *Proceedings of IEEE INFOCOM*, 2006.