Brief Announcement: Do VNet Embeddings Leak Information about ISP Topology?

Yvonne-Anne Pignolet[1], Stefan Schmid[2], and Gilles Tredan[3]

[1] ABB CRC, Switzerland
[2] TU Berlin & T-Labs, Germany
[3] CNRS-LAAS, France

Abstract. This paper initiates the study of adversarial topology inference with virtual network (VNet) embeddings in ISP networks. As an example, we sketch how to infer cactus graphs with VNet request complexity $O(n)$.

1 Contribution Sketch

An Internet Service Provider's (ISP) network infrastructure properties often constitute a business secret, not only for a competitive advantage, but also because the discovery of, e.g., bottlenecks, may be exploited for attacks or bad publicity. Hence, providers are often reluctant to open the infrastructure to novel technologies and applications that might leak information. We raise the question whether today's trend of *network virtualization* [1], can be exploited to obtain information about the infrastructure. Network virtualization allows customers to request *virtual networks (VNets)* on demand. A VNet defines a set of virtual nodes (e.g., virtual machines) interconnected via virtual links according to the specified VNet topology over a substrate network. In this paper we consider VNet requests which do not impose any location constraints on where the virtual nodes are mapped to. This flexibility in the VNet specification can be used by the operator to optimize the VNet embedding. Thus the VNet can be realized on arbitrary substrate nodes and paths. We assume that as long as a network provider has sufficient resources available to embed a VNet, it will always accept the request. We study how this behavior can be exploited to infer the *full topology* of the substrate.

Model. Our setting comprises two entities: a *customer* (adversary) issuing VNet requests and a *provider* that performs access control and embeddings of VNets (e.g., [2]). We model VNet requests as simple, undirected, weighted graphs $G = (V, E)$ where V denotes the virtual nodes and E denotes the virtual edges connecting nodes in V. Both sets can be weighted to specify requirements, e.g., computation or storage resources at nodes or bandwidth of edges. The infrastructure network (*substrate*) is a weighted undirected graph $H = (V, E)$, with V denoting the substrate nodes, E the substrate links, and the weights describe the capacity of nodes and edges. Without loss of generality, we assume that there are no parallel edges or self-loops either in VNet requests or in the substrate, and that H is connected. In order to focus on topological aspects, we assume the substrate graph elements in H to have a constant capacity of one unit and the requested nodes and links to come with a demand of one unit as well. A virtual link which is mapped to more than one substrate link, i.e., forms a *path*, uses resources of $\epsilon > 0$ at the *relay nodes*, the substrate nodes which do not constitute endpoints of the virtual link and merely serve for forwarding. As a performance measure, we introduce the notion of *request complexity*, i.e., the number of VNet requests which have to be issued until a given network is fully known to the adversary. Thus we study algorithms that "guess" the substrate topology H among the set $\mathcal{H}$ of possible topologies allowed by the model. Given a VNet request G, the provider always responds with an *honest binary reply* informing the customer whether the requested VNet is embeddedable on the substrate. Hence we assume that the provider does not use any means to conceal its network, e.g., by randomizing its binary replies. Based on the reply, the customer may then decide to ask the provider to embed the corresponding VNet G on H, or to continue asking for other VNet embeddings.

Cactus Graph Inference Algorithm. *Cactus graphs* are particularly interesting in the networking context (cf e.g., *Rocketfuel networks*, `www.cs.washington.edu/research/networking/rocketfuel/`). Formally, a cactus is a connected graph in which any two simple cycles have at most one node in common. Or equivalently, every edge in the cactus graph belongs to at most one 2-connected component, i.e., cactus graphs do not contain diamond graph minors. The cactus discovery algorithm CACTUS is based on the idea of incrementally growing the request graph and adding longest sequences of cycles (in our case triangle graphs consisting of three virtual nodes, short: Y) and chains (in our case two connected virtual nodes, short C) recursively. We first try to find the basic "knitting" of the "branches" of the given cactus. Only once such a maximal sequence is found for a branch, the algorithm discovers the detailed structure of the chain/cycle sequence by inserting as many nodes on the chains and cycles as possible. Intuitively, the nodes of a longest sequence of virtual cycles and chains will serve as an "anchor" for extending further branches in future requests: since the sequence is maximal and no more nodes can be embedded, the number of virtual nodes along the sequence must equal the number of substrate nodes on the corresponding substrate path. The endpoints of the sequence thus cannot have any additional neighbors, and we can recursively explore the longest branches of nodes discovered along the sequence.

Theorem 1. CACTUS *discovers any cactus with optimal request complexity $\Theta(n)$.*

Proof Sketch: The correctness of the algorithm follows from the fact that requesting a cyclic "motif" Y saturates a corresponding 2-connected component of the cactus graph, and k consecutive triangles (triangles having one common vertex, Y^k) can only be embedded on ℓ consecutive 2-connected components if $k \leq \ell$ (i.e., they constitute anchors). Once we have identified the maximal k, j such that $(Y^k C)^j$ can be embedded in H, we know that each of these Y^k motifs capture the basic "knitting" of the part of the cactus branch and for each C of this knitting the next requests find the maximal k', j' to replace it by $C(Y^{k'} C)^{j'}$. The time complexity then follows from assigning request costs to the cactus edges. The lower bound and asymptotic optimality is due to the number of possible cactus graphs and the binary reply scheme. □

References

1. Chowdhury, M.K., Boutaba, R.: A survey of network virtualization. Elsevier Computer Networks 54(5) (2010)
2. Even, G., Medina, M., Schaffrath, G., Schmid, S.: Competitive and Deterministic Embeddings of Virtual Networks. In: Bononi, L., Datta, A.K., Devismes, S., Misra, A. (eds.) ICDCN 2012. LNCS, vol. 7129, pp. 106–121. Springer, Heidelberg (2012)