The Effects of Topologies on the Performance of Real-Time Networks

Philip Diederich*, Alexej Grigorjew§, Stefan Geißler§, Tobias Hoßfeld§, Wolfgang Kellerer*,
*Technical University Munich, Germany; §University of Würzburg, Germany
{philip.diederich, wolfgang.kellerer}@tum.de
{alexej.grigorjew, stefan.geissler, tobias.hossfeld}@uni-wuerzburg.de

***Abstract*—Time-sensitive applications are increasingly prevalent in various network domains, such as industrial, medical, and vehicular communications, imposing substantial demands on network infrastructure. Consequently, ensuring low latency has become a crucial requirement for future networks, particularly through the implementation of deterministic latency network controllers. However, it is essential to recognize that the network controller represents just one facet of network performance management. The configuration of the network's topology also significantly influences its overall performance. This study, therefore, investigates the impact of different topologies on network performance, specifically focusing on deterministic latency guarantees. Our analysis shows the correlation between graph metrics characterizing the topology and its performance. This correlation facilitates a straightforward ranking of topology performance during critical phases like network planning or expansion. We introduce a readily obtainable graph metric that enables relative performance ranking without the need for exhaustive simulations or emulations. The metric exhibits a Spearman Ranking correlation coefficient exceeding 0.93.**

***Index Terms*—deterministic networking, graph theory, parameter study**

I. Introduction

In a time marked by the pursuit of faster, more reliable, and efficient data transmission, Time-Sensitive Networking (TSN) has emerged as a pivotal paradigm, revolutionizing the landscape of communication and networking technologies. The increasing integration of real-time applications, such as industrial automation and autonomous vehicles, demands a level of precision that conventional networking protocols struggle to achieve. TSN addresses these challenges by providing a standardized framework for time-critical communication, ensuring deterministic and low-latency data delivery. As industries transition towards automation and interconnected systems, the significance of TSN in ensuring temporal accuracy and reliability has become paramount.

Such latency-critical applications, typically operated in campus networks, depend on multiple aspects covered by the real-time network paradigm that TSN describes, such as fine-granular control of infrastructure components, precise monitoring, and the reservation of resources to realize the required latency requirements on top of a shared, physical infrastructure. In these campus deployments, a single provider controls and manages the entire network, from planning, to provisioning, and finally operation.

An efficient topology plays a crucial role in the effectiveness of real-time operations, specifically when it comes to the reservation of streams required for the stringent requirements imposed by time-critical applications. The choice of topology directly influences factors such as packet delivery time, network utilization, and overall performance, which are critical considerations for ensuring the successful reservation of streams. In a TSN network, a well-designed topology helps to optimize the allocation of network resources, preventing congestion, and providing the necessary bandwidth for real-time traffic. Furthermore, TSN deployments frequently involve the coexistence of different types of traffic, including both time-sensitive and non-time-sensitive data. An efficient topology allows for effective segregation and prioritization of traffic, ensuring that streams with stringent timing requirements are not adversely affected by less time-critical data. This segregation is crucial for maintaining the determinism and reliability necessary for TSN applications to function as intended.

Existing systems for predictable latency such as Chameleon [1] or Silo [2] ensure that real-time flow requirements like a flow's absolute deadline and minimum rate are met. Typically, these systems monitor the network state on a global scale. This centralized view enables them to decide if an additional flow request can be admitted to the network or if embedding a new flow leads to a violation of real-time requirements. However, those state-of-the-art systems for predictable low latency are typically designed for data center networks following a strictly determined data center topology. In contrast, emerging real-time applications have to run in arbitrary network topologies, unlike hierarchically structured data center topologies.

In contrast, the study conducted in the context of this paper includes the explicit validation of strict delay requirements that either pass or fail, instead of only minimizing delay. In addition, we propose a simple graph metric that provides a strong correlation to the expected flow admission performance. Hence, in this paper, we aim to address the following research question: *Is it possible to compare the flow admission performance of two topologies based on their graph representation?* Understanding how different topologies impact the performance of flow admissions provides valuable information when planning or upgrading such networks. Specifically, to answer the posed research question, we make the following contributions in the scope of this work:

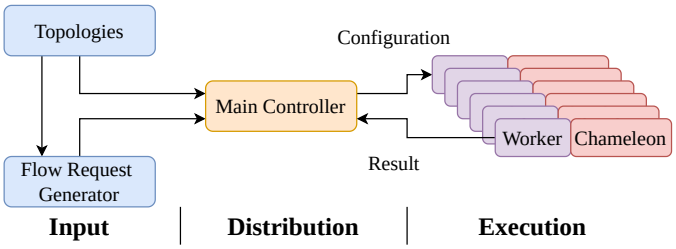


Fig. 1: Data flow overview for running the tests to get the performance of networks

- We perform a large-scale study to show the impact of various network topology parameters on the number of flow admissions with latency guarantees;
- We propose a graph metric that maps the expected performance in terms of latency guarantees to a given network topology;
- We show that our proposed metric is able to achieve a Spearman rank correlation coefficient of 0.93;

II. Related Work

The performance of networks depends on their underlying topology and employed technology. With Software Defined Networking (SDN), the authors of [3] conclude that processing times depend on the size of the network. Furthermore, [4] shows the dependency between the amount of control traffic and the network's size and network type. [5] obtains the correlation between different SDN performance indicators and graph metrics of the topologies. The knowledge can also be applied in the other direction to generate topologies with similar performance and features [6], [7]. Moreover, the authors of [8] find that the same graph metric can describe different datasets. The authors of [9] define a custom measure to rank a topology's ability to ensure low latency in changing conditions under different routing schemes. To the best of our knowledge, nobody obtained a correlation between the performance of deterministic network controller and graph metrics of the underlying topology.

III. System Overview and Methodology

A. Architecture

Network Controller: At the core of the system is the deterministic network controller. One feature of such controllers is that they guarantee latency for embedded flows. Controllers that provide guaranteed latency already exist. Three deterministic network controllers are: Chameleon [1], Silo [2], and QJump [10]. This paper chooses Chameleon as its network controller since it adds reconfiguration as a feature. Chameleon uses resource allocation and reservation to check if an admission of a flow is possible. Furthermore, it uses a greedy routing strategy for choosing paths. However, reconfiguration allows the Controller to re-evaluate and re-embed previously embedded flows to potentially embed more flows [1]. When using Chameleon, the flows need to follow arrival curves that can be described with the Token-Bucket model [11], [12].

System Design: Figure 1 shows the system's architecture. The system consists of three main stages: input (Topology and Flow request Generator), distribution (main controller), and execution (Worker-Chameleon combinations). The input stage provides the main controller with topologies and flow requests. Then, the main controller generates jobs and distributes them to the Worker-Chameleon combinations. The Worker-Chameleon combination executes the test case and stores the results. The separate input and distribution stages function as administration for the parallel execution of multiple Worker-Chameleon combinations, where each Worker-Chameleon instance functions as a standalone Chameleon controller to execute a single test case. This also allows for parallelization on multiple servers. The following section describes the workflow in detail.

B. Methodology

The system presents Chameleon with topologies and flow requests to obtain a network's performance. The following section describes the process of obtaining the number of successful embeddings - the performance metric - in detail. It also highlights what settings the user can change.

First, the user selects a set of topologies and a Flow Request generator via configuration. The user can choose between single-flow and categorical requests and specify the parameters for the chosen request generator. The system forwards the information to the distribution stage. The main controller combines each topology with all possible flow request settings. The combination of a topology and settings for the flow request generator is called a job. The distribution stage iteratively generates all possible jobs from the configuration. Next, the main controller sends the job to an available Worker-Chameleon combination (worker). Worker-Chameleon combinations are self-contained units that allow the parallel execution of multiple test cases. The worker receives the settings for the test run and sends the topology to Chameleon. Chameleon initializes with the topology and notifies the worker when ready. After the worker receives the signal from Chameleon, it generates the first admission request. The flow generation depends on what Flow Request Generator the user chooses.

1) **Single-flow requests:** All admission requests are the same in this option. The user controls the flow specification with the generator settings. Table II describes all flow parameter ranges from the evaluation. The worker receives a flow request generator with a single flow specification. Here, only the source and destination of the flow request change between requests. When the worker asks for a new admission request, the single-flow generator returns the current flow specifications and samples a source and destination. Part 3) describes the source-destination sampling.

2) **Categorical flow requests:** This option has different specifications for each flow. Here, the generator has four categories that specify ranges for the flow parameters. The user

can specify the distribution of categories in the configuration. Then, the generator samples a category. From the category, the generator gets upper and lower bounds for burst, rate, and deadline. Next, the generator samples burst, rate, and deadline uniformly at random from the ranges. This process is the same as the authors van Bemten et al. describe in [1]. Finally, the generator samples the source-destination pair, which Part 3) describes.

3) **Source and destination:** The request generator selects the source-destination pair uniformly at random from all host pairs, where source and destination are not the same.

The admission request combines the source-destination pair with the flow specification. Next, the worker sends the admission request to Chameleon. Chameleon tries to embed the flow request into the current state of the network. If Chameleon succeeds, it sends the path for the flow back. If Chameleon does not find a path directly where all flow requirements of currently embedded flows are met and the requirements of the new flow are met, it tries to reroute previously embedded flows that share the same links as the new flow request. The authors van Bemten et al. describe the process in more detail in [1]. Chameleon tries up to ten reroutings. If the reroutings do not result in a valid path for the new flow, Chameleon sends no route back to the worker. The system considers this a failure. When a failure occurs, the worker discards the flow request and continues with the next. To ensure a more uniform utilization of the network, the system allows for 100 failures. The failures can be consecutively or distributed during the embedding process. The worker has a failure counter, and if that failure counter reaches 100, the test case terminates, and then the worker saves the results. This concludes the execution of one job. After this, the worker receives the next job with new parameters and repeats the execution.

Topology Selection: This paper uses three types of topologies for the test cases. First, Barabási-Albert graphs [13] with a different number of nodes and the initial node degree ranging from one to four. Barabási-Albert graphs are scale-free networks where nodes are added using preferential attachment [13]. Second, graphs from the Internet Topology Zoo Dataset [14], which is a collection of real-world networks created from public information of these networks. Third, generated WAN-like topologies. The tool from Dietz et al. [7] uses an initial topology to generate more topologies that are similar to the initial topology. The initial topology for this paper's use-case is Agis from the Topology Zoo [14]. The paper uses 160 topologies from each category. Furthermore, Table I summarizes important parameters. The system interprets the topologies as a combination of switches and links, i.e., each node is a switch, and each edge is a link. The test system connects four hosts to each switch to have enough sources and destinations to ensure that the network bottlenecks and not a connection between a host and a switch. Furthermore, all links have the same link speed of 1Gbit/s. Higher link speeds result in more flow admission but also in longer execution times.

TABLE I: Overview of Topologies

Category	Parameters
Barabási-Albert	Nodes: 15-34; Edges: 14-120 Initial Node Degree: 1-4
Topology Zoo	Nodes: 4-37; Edges: 4-76
Generated Topologies	Nodes: 25; Edges: 24-41 Initial Topology: Agis

TABLE II: Settings for the single-flow evaluations

Parameter	Values
Burst [Bytes]	100, 500, 900, 1300
Rate [Mbit/s]	0.1, 0.4, 0.7, 1.0, 4.0, 7.0, 10.0
Deadline [ms]	10, 50, 90

C. Topology Metric

The evaluation in this paper relates the performance of the network with its graph metrics. Here, the performance of the topology is measured as successful admissions in the network. Easy to obtain graph metrics of a graph $G = (V, E)$ are the number of nodes $|V|$ and the number of edges $|E|$. Furthermore, this paper looks at the closeness centrality [15] in its normalized form. The normalized form is defined as:

$$C_N(x) = \frac{|V| - 1}{\sum_{y \in V} d(x, y)} \tag{1}$$

where $d(x, y)$ defines the shortest path length between node x and y.

This paper proposes the custom metric:

$$M(G) = |E| \cdot min(C_N(G)) \tag{2}$$

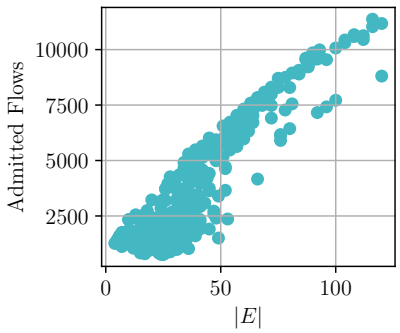
where $C_N(G)$ defines the normalized closeness centrality for all nodes in G. The custom metric combines the number of edges $|E|$ with the minimal normalized closeness centrality of the nodes by multiplication. Section IV-A1 explains the reasoning behind this metric.

IV. Evaluation

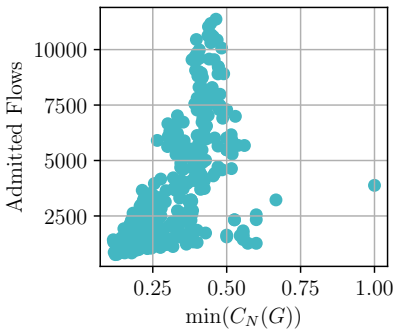
This evaluation uses the system described in III. The evaluation uses 480 different topologies and the two types of flow request generation. Each topology is combined with all different flow request settings. This results in over 590.000 different test cases. In all test cases, the performance, i.e., the number of successful admission was obtained with Chameleon.

A. Evaluation with 480 different Topologies

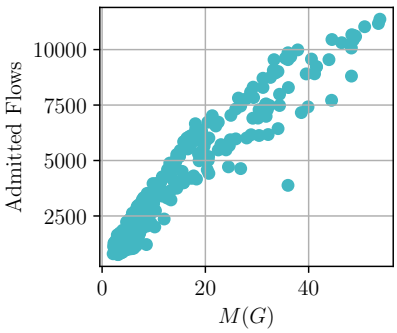
1) Ranking Relative Performance: The main goal of this evaluation is to rank topologies according to their performance based on their respective graph metric. The evaluation aggregates the results of all test cases for each topology and ranks them based on the aggregate. Furthermore, the evaluation uses the Spearman rank correlation coefficient to compare the performance of different metrics. The Spearman rank correlation coefficient measures the ranking performance of the metrics. The coefficient ranges from -1 to $+1$, where -1 means perfect reverse ranking and $+1$ means perfect ranking. Figure 2 shows the ranking results for three different metrics.



(a) Number of Edges as metric. Spearman ranking correlation of 0.7987



(b) $min(C_N(G))$ as metric. Spearman ranking correlation of 0.7218



(c) $M(G)$ as metric. Spearman ranking correlation of 0.9317

Fig. 2: Using three different metrics to correlate the number of successful admissions with topologies

First, Figure 2a ranks the topology's performance, i.e., the number of successful flow admissions based on the number of edges in the topology. Second, Figure 2b compares the ranking based on the topology's minimal closeness centrality and performance. Third, Figure 2c uses the custom metric $M(G)$ from Sec. III-C to rank the performance.

Using $|E|$ as the metric allows for good rankings with topologies with more edges ($|E| \geq 60$). However, the metric fails to rank the topologies with fewer edges effectively. The problem with this metric is that many different topologies with the same number of edges exist. Each of these topologies can have the same number of edges and vastly different performance. For example, a full mesh topology with 5 nodes has 10 edges. A star topology with 11 nodes also has 10 edges. However, both topologies have different performance. Overall, using $|E|$ as a metric achieves a Spearman rank correlation coefficient of **0.7987**.

Next, we take a look at the minimal closeness centrality as a metric to rank the performance. The minimal normalized closeness centrality can take values from 0.0 to 1.0 regardless of the network's size. This results in rather bad rankings. For example, considering only full mesh topologies, the minimal closeness is always 1.0. However, a topology with 10 nodes has more admitted flows than a topology with only 4 nodes. Therefore, the minimal closeness alone is insufficient to rank the topologies relatively. The minimal closeness centrality achieves a Spearman rank correlation coefficient of **0.7218**.

The metric $M(G)$ from Sec. III-C combines both previous metrics. With the dataset used, it achieved the best ranking performance. $M(G)$ uses the minimal closeness centrality as a discounting factor. This factor can account for the structural differences in the topologies. Accounting for structural differences helps when the topologies have the same number of edges. For example, a star topology with 11 nodes and a full mesh with 5 nodes have 10 edges each. Here, the discount factor for the star topology is 0.53 (the minimal closeness centrality in the star topology) and 1.0 for the full mesh. However, $M(G)$ not only considers the structure but also the size of the network. For example, for full-mesh topologies, the discount factor is always 1.0. Then, the ranking relies on the number of edges $|E|$ to rank the topologies. The metric $M(G)$ achieves a Spearman rank correlation coefficient of **0.9317**

Importantly, $M(G)$ can only rank the topologies relative to each other. $M(G)$ can not make statements on absolute performance. The absolute performance largely depends on inputs like the flow request that the controller has to embed. Furthermore, $M(G)$ only delivers valid results for scenarios when the input flow requests are similar.

2) Differences in $M(G)$: The metric is not intended to estimate the absolute number of flows the network can embed but to rank topologies. The ranking makes it easier to choose better-suited topologies when needed. Therefore, the question arises: If we have two different topologies, at what difference in $M(G)$ is the performance difference statistically significant? To answer this, we look at the effect differences in $M(G)$ have on performance. We calculate the difference of every pair of points for $M(G)$ and the number of admitted flows.

Figure 3 shows the results for all differences in $M(G)$ in the data set. The blue error bars contain the confidence interval with confidence of 95%. The subplot in Figure 3 highlights the results in the range 0 to 10. Figure 3 can be interpreted as follows: If two topologies have a difference of five in $M(G)$, we see that the flow admission performance differs in the mean by 1800 flows. However, the absolute value depends largely on the parameters of the flows. The confidence intervals that are colored blue do not overlap, which means that the performance difference is statistically significant with a confidence of 95%. In particular, the confidence intervals do not overlap starting with $\Delta M(G)$ of 1. Hence, for differences of $M(G) < 1.0$, we cannot make a definite decision on which topology is better. For the data set, this also mostly works with the difference in the number of edges. However, not with the difference in the minimal normalized closeness centrality.

B. Discussion

The results the evaluation achieves are based on assumptions. The set of topologies plays a role. We try to provide a wide variety of different topologies. Here, we assume that we have equal link speeds on all links in the topology. This limits the comparability of the metric. The goal is to address this issue in future work. Furthermore, the sources and destinations

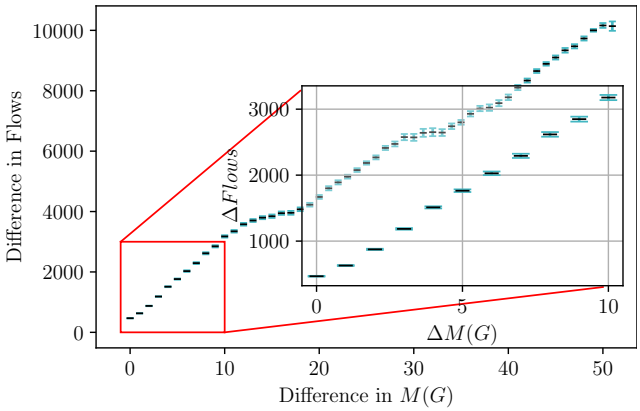


Fig. 3: The effect that a difference in $M(G)$ has on the performance of the topologies with a confidence of 95%.

are uniformly distributed among all possible hosts. In future work, we want to address different demand distributions.

The data were created with a particular combination of flow request settings, topologies, and controller settings. Therefore, it is very likely that our custom metric is only valid for that particular setup. We acknowledge that changing the controller to a different controller might require a different metric since resource allocation and routing strategies of controllers can be different. However, the idea of a correlation between the topology and performance should still hold. The evaluation considers the number of successful admissions as the network performance metric. Changing the performance metric, e.g. to network utilization, might require a different metric.

Furthermore, the metric this paper presents is meant to rank topologies relative to each other. It will not provide accurate results for the absolute number of embedded flows since the number of embedded flows largely depends on the flow requests. Therefore, the idea of finding a metric that allows for quick relative ranking of topologies also holds for different settings and scenarios. However, the metric might be different.

V. Conclusion and Future Work

Real-time networks gain more popularity with the advance of time-sensitive applications like industrial automation and autonomous vehicles. The network's ability to admit flows while keeping the real-time demands of all other flows is crucial. Here, the topology plays a critical role. However, choosing the right topology in the planning phase or changing an existing topology to account for more flows is time-consuming since all qualified topologies need to be tested.

This paper explores the connection between the topology and flow admission performance via a custom metric to quickly rank topologies. We show that our metric performs better than other metrics like the topology's size alone. With this easy-to-obtain metric, topologies can be quickly ranked relative to each other and narrow the number of possible topologies when planning a real-time network. Our custom metric achieved a Spearman rank correlation of 0.93.

In the future, we want to further use the knowledge about the correlation between the network's flow admission performance and its representation as a graph to help plan better-performing networks for real-time use cases. For example, the proposed metric could be used to judge possible changes to a network faster than testing the changes with the network controller.

Acknowledgement

This work received funding by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) 316878574.

References

[1] A. Van Bemten, N. Derić, A. Varasteh, S. Schmid, C. Mas-Machuca, A. Blenk, and W. Kellerer, "Chameleon: predictable latency and high utilization with queue-aware and adaptive source routing," in *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*, CoNEXT '20, (New York, NY, USA), pp. 451–465, Association for Computing Machinery, Nov. 2020.

[2] K. Jang, J. Sherry, H. Ballani, and T. Moncaster, "Silo: Predictable Message Latency in the Cloud," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, (New York, NY, USA), pp. 435–448, Association for Computing Machinery, Aug. 2015.

[3] C. Metter, S. Gebert, S. Lange, T. Zinner, P. Tran-Gia, and M. Jarschel, "Investigating the impact of network topology on the processing times of SDN controllers," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, (Ottawa, ON, Canada), pp. 1214–1219, IEEE, May 2015.

[4] M. Z. Naseer and V. Fodor, "The effect of network topology on the control traffic in distributed SDN," in *2018 IFIP Networking Conference (IFIP Networking) and Workshops*, (Zurich, Switzerland), pp. 199–207, IEEE, May 2018.

[5] N. Gray, K. Dietz, and T. Hossfeld, "Simulative Evaluation of KPIs in SDN for Topology Classification and Performance Prediction Models," in *2020 16th International Conference on Network and Service Management (CNSM)*, pp. 1–9, Nov. 2020. ISSN: 2165-963X.

[6] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "Network topology generators: degree-based vs. structural," *ACM SIGCOMM Computer Communication Review*, vol. 32, pp. 147–159, Aug. 2002.

[7] K. Dietz, M. Seufert, and T. Hoßfeld, "Want More WANs? Comparison of Traditional and GAN-Based Generation of Wide Area Network Topologies via Graph and Performance Metrics," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2023. Conference Name: IEEE Transactions on Network and Service Management.

[8] MahadevanPriya, KrioukovDmitri, FomenkovMarina, DimitropoulosXenofontas, c. c, and VahdatAmin, "The internet AS-level topology," *ACM SIGCOMM Computer Communication Review*, Jan. 2006. Publisher: ACM PUB27 New York, NY, USA.

[9] N. Gvozdiev, S. Vissicchio, B. Karp, and M. Handley, "On low-latency-capable topologies, and their impact on the design of intra-domain routing," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, (Budapest Hungary), pp. 88–102, Association for Computing Machinery, Aug. 2018.

[10] M. P. Grosvenor, M. Schwarzkopf, I. Gog, R. N. M. Watson, A. W. Moore, S. Hand, and J. Crowcroft, "Queues Don't Matter When You Can JUMP Them!," pp. 1–14, 2015.

[11] J.-Y. Le Boudec and P. Thiran, *Network calculus: a theory of deterministic queuing systems for the Internet*. No. 2050 in Lecture notes in computer science, Berlin ; New York: Springer, 2001.

[12] A. van Bemten and W. Kellerer, "Network Calculus: A Comprehensive Guide," Technical Report 201603, Technical University Munich, Munich, Aug. 2016.

[13] A.-L. Barabási and R. Albert, "Emergence of Scaling in Random Networks," *Science*, vol. 286, pp. 509–512, Oct. 1999. Publisher: American Association for the Advancement of Science.

[14] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, pp. 1765–1775, Oct. 2011.

[15] A. Bavelas, "Communication Patterns in Task-Oriented Groups," *The Journal of the Acoustical Society of America*, vol. 22, pp. 725–730, June 2005.