UDC 004.72
Pestov O.[1], Kyrychek H.[2]
[1] student of gr. CNS-513m NU «Zaporizhzhia Polytechnic»
[2] PhD(Technology), assistant professor NU «Zaporizhzhia Polytechnic»

YGGDRASIL ROUTING SCHEME AS A BASIS FOR LARGE-SCALE DECENTRALIZED MESH NETWORKS

Nowadays centralized software-defined networking is becoming increasingly popular with Internet service providers due to its remarkable benefits in automated remote hardware configuration, monitoring, control, ease of deployment, use of «whitebox» hardware, etc. That said, its benefits come with a trade-off at the cost of network reliability and disruption resilience, as centralization adds singular points of failure to the system. A consequence of this is the relatively recent infamous Kyivstar network collapse [1] that occurred on December 12, 2023 resulting in several days of downtime with complete lack of customer connectivity. Numerous services throughout the country relied on Kyivstar and could not function until the damage was eventually mitigated. This incident shows how unreliable centralized hierarchical communication systems may be and calls for a different approach – mesh networking, where the line between routing and terminating (end) devices becomes blurry. The concept of mesh networking is not new, a number of routing protocols have been developed [2, 3, 4] and successfully used (e.g. the Freifunk initiative community networks [5]). However, there is yet to be one that would scale beyond small local networks and do so efficiently, as in use the least amount of system resources needed for functioning.

The goal of this work is to research how fitting the Yggdrasil routing scheme is for deploying large-scale decentralized communication networks. The research object is the implementation of the experimental Yggdrasil routing scheme [6]. The research subject is the scalability of Yggdrasil regarding hop limit, CPU usage and memory footprint of the routing daemon. The main point of the Yggdrasil routing scheme is in the use of key-based routing and a self-arranging spanning tree of the network, where at any given point in time the root of the tree is the node with the lowest key value, and keys in question being randomly generated ed25519 public keys. Route lookups are done on-demand using broadcasts. Nodes automatically peer with TCP connections over IPv6 using link-local addresses or with any other explicitly specified TCPv4/6 connections. Described approach has a number of advantages: nodes generate addresses for themselves independently; asymmetric end-to-end encryption with the same keys used for addressing; parts of a larger network can be linked through any other IPv4/6 networks the nodes participate in and impossibility of loops in the spanning tree.

The benchmarking was conducted using meshnet-lab [7] environment with custom system resource monitoring scripts due to its flexibility, possibility of

simulating large-scale topologies with Linux network namespaces and automating the process with Python. It should be noted, however, that the host system the tests were conducted on can't handle more than 750 nodes at once. Also, as the worst case scenario, only line topology will be considered in this work. Packet arrival over time for different network sizes on line topology is featured on figure 1.

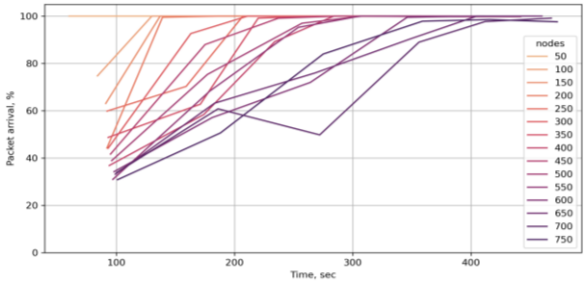

Figure 1 – Packet arrival over time for different network sizes, line topology

Another experiment was conducted to look into the difference in CPU usage for transmitting, receiving and intermediate nodes. On a line topology of 50 nodes traffic was generated between the first and last one using the iperf3 utility [8]. The experiment was conducted twice, with 100 Mbit/s links and unlimited bandwidth (as in not artificially limited by the environment). The results, featured in figures 2 and 3 respectively, confirm the assumption about receiving and transmitting nodes having the most intensive CPU usage, with intermediate node readings rising the closer they are to the receiving node (hue represents node number).

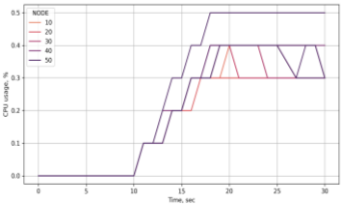

Figure 2 – CPU usage over time, line topology, 50 nodes, 100 Mbit/s links

Lastly, an experiment was conducted on a 750 node line topology with slow links (10 Mbit/s bandwidth, 10±5 ms latency). The experiment showed 749 hop communication to be possible still, albeit with a considerable latency (18 to 20 seconds), which calls for delay tolerant applications to be used with such a network.

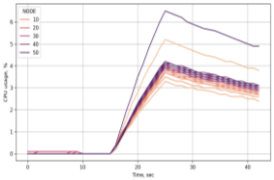

Figure 3 – CPU usage over time, line topology, 50 nodes, unlimited bandwidth

Overall, the Yggdrasil routing scheme looks like a promising basis for deploying large-scale self-configuring communication networks. However, before planning such a network, a number of additional topics must be researched, like node mobility testing, functioning in realistic latency and packet loss conditions, preferred underlying link layer technologies and required hardware, deployment on common hardware, etc.

REFERENCES

1. Collier K., Grudinin A. Ukraine's top mobile internet company says it has been hit by Russian cyberattack // NBC News. [Electronic resource]. – Mode of access: https://www.nbcnews.com/tech/security/ukraines-top-mobile-internet-company-blames-russian-cyberattack-rcna129253 (date of access: 05.04.2024).

2. B.A.T.M.A.N. [Electronic resource] // Open-Mesh. – Mode of access: https://www.open-mesh.org/projects/open-mesh/wiki (date of access: 05.04.2024)

3. Clausen T., Jacquet P. RFC3626: Optimized link state routing protocol (OLSR). - 2003. at: http://www.researchgate.net/publication/270394473.

4. Chroboczek J., Schinazi D. RFC 8966: Babel Routing Protocol, 2021. DOI: https://doi.org/10.17487/RFC8966.

5. Freifunk [Electronic resource] // freifunk.net. – Mode of access: https://freifunk.net/en/ (date of access: 05.04.2024).

6. Yggdrasil Network [Electronic resource] // Yggdrasil Network. – Mode of access: https://yggdrasil-network.github.io/ (date of access: 05.04.2024).

7. GitHub - mwarning/meshnet-lab: Emulate huge mobile ad-hoc mesh networks using Linux network namespaces. [Electronic resource] // GitHub. – Mode of access: https://github.com/mwarning/meshnet-lab (date of access: 05.04.2024).

8. iPerf - The ultimate speed test tool for TCP, UDP and SCTP. [Electronic resource]. – Mode of access: https:// https://iperf.fr/ (date of access: 05.04.2024).