Low-Load Distributed Data Networks: Concept, Classification and Comparative Analysis

Author: Kolpakov Viktor

Related: Zero Knowledge Infrastructure manifesto, comparison of networks.

Abstract

This article introduces the concept of a *low-load distributed data network*: a network in which the load on each node (traffic, storage, service state) is bounded by that node's own circle of participants and does not grow as the network grows. We show that the load classifications used in communications theory (utilization, erlangs, the "knee" and "cliff" points on the throughput-versus-load curve, quality-of-service classes) describe the *state* of a network at a given moment rather than a property of its architecture, and therefore cannot distinguish networks by this trait. We propose six traits of a low-load network and a classification LL-0…LL-3 by the asymptotic load on a node. We compare fifteen networks, from FidoNet (1984) to Nevod (2026), and name their predecessors in the literature: friend-to-friend (F2F) networks, delay-tolerant networks (DTN), distributed hash tables, local-first software and community networks.

Keywords: distributed networks, decentralization, per-node load, scalability, F2F, DTN, peer-to-peer networks, messengers, Zero Knowledge Infrastructure.

---

1. Introduction

Most of today's messaging services are centralized: the whole network is served by one company's servers, and the load on those servers grows with the number of users. Decentralized alternatives (federated networks, peer-to-peer overlays, anonymity networks) distribute *ownership* of the infrastructure but, as shown below, usually not the *load*: a federation server or overlay node carries traffic, history and service state produced by other people's users, and its load again grows with the size of the network or of its popular part.

The aim of this work is to single out the class of networks in which a node's load is determined only by its own circle: the users and software agents (bots, AI agents, services) for which it is the home node; to define this class, to propose a measurable classification and to compare existing networks by this trait.

2. Load in data networks: existing concepts

2.1 Utilization

In queueing theory the load of a channel or node is characterized by the utilization factor $\rho = \lambda/\mu$, the ratio of the arrival rate to the service rate [1]. As $\rho \to 1$, queue length and delay grow without bound; at $\rho > 1$ the system is overloaded. In practice rough levels are used: low utilization ($\rho <$ ~0.3), moderate (0.3–0.7), high (0.7–0.9), saturation and overload; operators usually plan links with headroom up to ~70–80% at peak hour. These thresholds are engineering guidelines, not a single standard.

2.2 Teletraffic: erlangs and the busy hour

In telephony, traffic intensity is measured in erlangs (1 Erl is one channel occupied continuously) [2]; dimensioning is based on the busy hour; offered, carried and lost traffic are distinguished, and the grade of service is set as an acceptable blocking probability (Erlang B and C formulas) [1, 2].

2.3 How a network reacts to growing load: the "knee" and the "cliff"

Jain, Ramakrishnan and Chiu [3] described three operating regions of a network: before the "knee", throughput grows with load at low delay; between the "knee" and the "cliff", throughput barely grows while delay rises quickly; past the "cliff", congestion and collapse set in. At the "knee" the ratio of throughput to delay (the network's "power") is at its maximum; this distinction underlies congestion avoidance (keeping operation near the "knee") and congestion control (preventing a fall off the "cliff"). TCP congestion control is built on these ideas [4].

2.4 Traffic and quality-of-service classes

The Differentiated Services architecture [5] introduces per-hop behaviors (Expedited Forwarding EF, Assured Forwarding AF, Best Effort BE); ITU-T Recommendation Y.1541 [6] defines IP network QoS classes 0–5 by delay, jitter and loss. These are classifications of *requirements* for delivery, not of load.

2.5 Load regimes: from underload to collapse

Network operating regimes by load are described in standards and classic works:

Regime	Where defined	Meaning
Underload (underutilization)	descriptive, not defined in standards	links are noticeably idle; often a deliberate capacity reserve (overprovisioning)
Normal load	ITU-T E.500 [36]	frequent busy periods in which user expectations should be met; target grade-of-service values are set for it
High load	ITU-T E.500 [36]	infrequent peak periods: expectations may not be met, but performance must stay high enough to keep congestion from spreading through the network
Overload	ITU-T E.412 [37]	traffic exceeds the network's capacity; management controls kick in: automatic congestion control, circuit reservation, rerouting
Congestion collapse	RFC 896, J. Nagle, 1984 [38]	every packet is sent many times and useful throughput drops almost to zero; first observed in October 1986, when the NSFnet backbone fell from 32 kbit/s to 40 bit/s

There is no established term for a "fully loaded" network; the level a network is dimensioned for corresponds to "normal load" in E.500.

All these regimes are **states** of a network at a given moment: the same network is underloaded in the morning, at normal load in the evening and overloaded during an outage. Load regimes and the architectural classes introduced below are independent of each other: a network of class LL-0 can also become overloaded, for example if a node sits on a weak link.

2.6 Scaling of per-node load

In the distributed-systems literature a node's cost is assessed asymptotically as a function of the number of participants N: broadcasting to all nodes costs O(N) per message; distributed hash tables (Chord [7], Kademlia [8]) cost O(log N) for routing and the neighbor table. There is also the claim that in peer-to-peer systems the maximum load per node does not grow with the number of nodes, because new nodes bring more resources than they consume [9], but this refers to *statistical balancing* of a load made up of other people's requests.

2.7 Conclusion

Classifications 2.1–2.5 answer the question "how loaded is the network *now*"; by them the same network is sometimes lightly and sometimes heavily loaded. For this reason the term "low-load network" in this article does not mean a network that is lightly loaded at the moment, but a network whose architecture bounds the load on each node. Only the approach of section 2.6 (how the load on a node grows with the size of the network) concerns a *property of the architecture*, and there is no standard classification of networks by this property.

3. Distributed networks: an overview of approaches

Centralized services (WhatsApp, 2009; Telegram, 2013; Signal, 2014 [10]): end-to-end encryption of content can be strong, but servers, accounts and metadata are held by one organization; server load grows with the entire audience.

Federated networks: e-mail, Matrix (2014) [11], ActivityPub/Mastodon (2016; W3C Recommendation 2018) [12]. Ownership is distributed among servers, but a server's load grows with its external connections: in Matrix the history of a large group chat (a "room" in Matrix terms) is replicated to every server that has at least one of its members, in ActivityPub delivery grows with the number of followers.

Peer-to-peer overlays and distributed hash tables: Freenet (2001) [13], Chord [7], Kademlia [8]. Nodes are equal, load is O(log N), but a node stores and routes data of strangers.

Friend-to-friend (F2F) networks. The term was proposed by D. Bricklin in 2000 [14]: a node connects only to the nodes of people it knows. Examples: Turtle (Popescu, Crispo, Tanenbaum, 2004) [15], OneSwarm [16], Freenet's "darknet" mode, RetroShare, GNUnet; storage on trusted nodes also belongs here [17]. A node's number of connections is bounded by its number of friends [18, 19], but messages travel along chains of friends through uninterested nodes, and routing in such networks remains a hard problem [18, 19, 20]. **F2F networks are the closest predecessor of the proposed concept**: they were the first to bound a node by its circle of acquaintances, but by the number of connections, not by load.

Anonymity overlay networks (onion routing): a development of D. Chaum's mix networks (1981) [39] and the onion routing of D. Goldschlag, M. Reed and P. Syverson (1996) [40]; implementations include Tor (2002–2004) [21] and Session (2020) [22]. Relay nodes run by volunteers or operators carry other people's transit in proportion to their bandwidth.

Delay-tolerant networks (DTN): Fall's architecture (2003) [23], RFC 4838 [24], Bundle Protocol v7, RFC 9171 [25]: "store and forward" across disrupted links. A historical implementation of the same principle is FidoNet (1984) [26], a modern one is NNCP (2016) [27].

Stacks for any medium and radio networks: Reticulum (protocol published in 2016) and the LXMF messenger [28]: operation from a few bits per second (LoRa, packet radio) up to the internet, encryption by default; Meshtastic (2020) [29]: text messaging over LoRa radio modules.

Community networks: Freifunk, Guifi.net [30]: networks created and run by their users themselves.

Local-first software [31] and **Secure Scuttlebutt** (2014) [32]: data lives with the user; in SSB a node stores the feeds of its friends and friends of friends, and synchronization happens when nodes meet.

Relays: Nostr (2020) [33]: simple relay servers, the popular ones loaded by the whole audience.

4. The concept of a low-load distributed network

4.1 Definition

A low-load distributed data network is a network in which each node serves a small, naturally bounded circle of participants and carries mostly their own traffic. Nodes are equal and run on ordinary hardware. The network transmits only what is necessary, tolerates delays and link disruptions, and has no center whose shutdown or seizure would stop it.

The concept concerns the *design* of a network, not its load at a given moment (sections 2.5, 2.7).

4.2 Traits

1. **Own circle**: a node's load grows with its circle, the people and software agents it serves, not with the size of the whole network. The main trait.
2. **Ordinary hardware**: a home computer, mini-server or router; no data center needed.
3. **Transmission economy**: text and signals first, bulky data on request.
4. **Delay tolerance**: with no connection, a message waits and is sent later, including over a slow channel.
5. **No center**: no single point of failure, blocking or seizure.
6. **Node blindness**: a node carries only ciphertext and cannot read it (the Zero Knowledge Infrastructure principle).

4.3 How it differs from its predecessors

Concept	What is bounded	How it differs
F2F [14–20]	a node's number of connections, by its circle of acquaintances	transit along chains of acquaintances remains; load is not bounded
DHT [7, 8]	load growth, by a logarithm	a node carries strangers' data
DTN [23–25]	nothing about load	about delays and disruptions, not load distribution
Local-first [31]	where data is stored: with the user	about data ownership, not the network
Low-load network	**a node's entire load, by its circle**	a combination of traits 1–6; trait 1 is the defining one

5. Classification: LL classes

The load class of a node is determined by the fastest-growing of its three components: traffic carried, data stored, service state maintained. N is the number of participants in the network, k is the size of the node's circle: the number of its users and software agents. For class LL-0 the load estimate contains only k, not N.

Class	Node load	Definition

LL-0	O(k)	the node carries, stores and remembers only what relates to its circle; no transit for others
LL-1	O(k) + bounded foreign	plus transit or storage for others, bounded independently of N (hop limit, quotas, neighbors chosen by the node itself)
LL-2	O(log N)	load grows slowly (distributed hash tables)
LL-3	O(N) or not bounded by its circle	load grows with the whole network or depends on how many outside users access the node (a popular server, a relay)

A low-load network in the strict sense is a network of class **LL-0**; LL-1 is almost low-load.

6. Comparative analysis

6.1 Traits (1 = met, ½ = partly, 0 = not met) and classes

Network	Year	Own circle	Hardware	Economy	Delays	No center	Blind node	Class
Secure Scuttlebutt [32]	2014	1	1	½	1	1	0	LL-0
Briar [35]	2018	1	1	½	½	1	1	LL-0
Nevod [34]	2026	1	1	½	½	1	1	LL-0
FidoNet [26]	1984	1	1	1	1	0	0	LL-1
NNCP [27]	2016	½	1	1	1	1	1	LL-1
Reticulum / LXMF [28]	2016	½	1	1	1	1	1	LL-1
Meshtastic [29]	2020	0	1	1	½	1	½	LL-1
Tor [21]	2002	0	½	0	0	1	1	LL-3
WhatsApp	2009	0	0	0	0	0	½	LL-3
Telegram	2013	0	0	0	0	0	0	LL-3
Matrix [11]	2014	0	½	0	0	½	½	LL-3
Signal [10]	2014	0	0	0	0	0	½	LL-3
Mastodon [12]	2016	0	½	0	0	½	0	LL-3
Nostr [33]	2020	0	½	0	0	1	½	LL-3
Session [22]	2020	0	½	0	½	1	1	LL-3

Notes. Signal and WhatsApp get ½ on trait 6: content is end-to-end encrypted, but the organization sees metadata and runs the servers. In Reticulum and NNCP intermediate nodes carry and store other people's packets (trait 1 = ½, class LL-1). Year is the first public release (Reticulum: in development since 2014; Signal: predecessors TextSecure/RedPhone, 2010; Briar: in development since 2011).

6.2 Observations

- By the main trait (the LL class) the strictest are Nevod, Briar and Secure Scuttlebutt (LL-0); Reticulum and NNCP belong to LL-1.
- All centralized and federated networks, as well as anonymity overlay networks, are LL-3: decentralizing ownership does not mean decentralizing load.

7. Nevod as an example of class LL-0, and its limitations

Nevod [34] is a messaging network built on the Zero Knowledge Infrastructure principle: a node serves only its own participants and carries encrypted messages without access to their content; there is no center; nodes run on home hardware or rented servers. The service table of nodes grows with the number of *nodes*, not users, and is small.

Limitations as of October 2026 (traits not fully met, not proven in practice):

- trait 3: media is transmitted whole, without previews or download on request;
- trait 4: undelivered messages are kept for 72 hours; works only over the internet;
- no forward secrecy: messages are encrypted to the recipient's long-term key;
- a node sees delivery metadata (who, to whom, when);
- a working prototype with no real users and no independent audit; iOS app only. The source code is open (AGPL-3.0) [41].

The traits were derived from Nevod's principles, so the comparison naturally favors it; Nevod's score reflects its architecture, not the maturity of the product.

8. Open questions

1. Storage for a recipient who is offline for a long time: who stores it and for how long, without breaking class LL-0.
2. Path finding between a large number of nodes without a central directory.
3. Is bounded transit (LL-1) acceptable for delivery over slow channels, and is reciprocity needed ("you carry others' traffic, others carry yours")?
4. Which components count as load: traffic, storage, state, energy use, administrator time.
5. A formal metric for a node's circle k and its measurement on real networks.

9. Conclusion

The load concepts of communications theory describe the state of a network, not a property of its architecture. The proposed concept of a low-load distributed network singles out an architectural class in which a node's load is bounded by its circle, and complements known approaches (F2F, DTN, distributed hash tables, local-first, community networks) with a combination of six traits. The LL-0…LL-3 classification makes the concept measurable and allows any network to be assigned to a class. Practical validation and a full literature review of closed databases (IEEE, ACM, eLibrary) are the next steps.

References

1. Kleinrock L. *Queueing Systems. Vol. 1: Theory.* — New York: Wiley, 1975.
2. Erlang A. K. The theory of probabilities and telephone conversations // Nyt Tidsskrift for Matematik B. — 1909. — Vol. 20. — P. 33–39.
3. Jain R., Ramakrishnan K. K., Chiu D.-M. Congestion Avoidance in Computer Networks with a Connectionless Network Layer. — DEC-TR-506, 1988. — https://arxiv.org/abs/cs/9809094

4. Jacobson V. Congestion Avoidance and Control // Proc. ACM SIGCOMM '88. — 1988. — https://ee.lbl.gov/papers/congavoid.pdf

5. Blake S. et al. An Architecture for Differentiated Services. — RFC 2475, 1998. — https://www.rfc-editor.org/rfc/rfc2475

6. ITU-T Recommendation Y.1541. Network performance objectives for IP-based services. — https://www.itu.int/rec/T-REC-Y.1541

7. Stoica I. et al. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications // Proc. ACM SIGCOMM 2001. — https://pdos.csail.mit.edu/papers/chord:sigcomm01/chord_sigcomm.pdf

8. Maymounkov P., Mazières D. Kademlia: A Peer-to-peer Information System Based on the XOR Metric // IPTPS 2002. — https://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf

9. LibreSocial: A Peer-to-Peer Framework for Online Social Networks. — 2020. — https://arxiv.org/abs/2001.02962

10. Signal Protocol: Technical Documentation. — https://signal.org/docs/

11. Matrix Specification. — https://spec.matrix.org/

12. Lemmer-Webber C., Tallon J. et al. ActivityPub. W3C Recommendation, 2018. — https://www.w3.org/TR/activitypub/

13. Clarke I., Sandberg O., Wiley B., Hong T. W. Freenet: A Distributed Anonymous Information Storage and Retrieval System // Designing Privacy Enhancing Technologies, LNCS 2009. — Springer, 2001. — https://link.springer.com/chapter/10.1007/3-540-44702-4_4

14. Bricklin D. Friend-to-Friend Networks. — 2000. — http://www.bricklin.com/f2f.htm ; overview: https://en.wikipedia.org/wiki/Friend-to-friend

15. Popescu B. C., Crispo B., Tanenbaum A. S. Safe and Private Data Sharing with Turtle: Friends Team-Up and Beat the System // Security Protocols, 12th International Workshop, 2004. — https://link.springer.com/chapter/10.1007/11861386_24

16. Isdal T., Piatek M., Krishnamurthy A., Anderson T. Privacy-preserving P2P data sharing with OneSwarm // Proc. ACM SIGCOMM 2010. — https://dl.acm.org/doi/10.1145/1851182.1851198

17. Li J., Dabek F. F2F: reliable storage in open networks // IPTPS 2006. — https://pdos.csail.mit.edu/~jinyang/pub/iptps-f2f.pdf

18. Evans N. S., GauthierDickey C., Grothoff C. Routing in the Dark: Pitch Black // ACSAC 2007. — https://www.researchgate.net/publication/4303478_Routing_in_the_Dark_Pitch_Black

19. Roos S., Strufe T. A Contribution to Analyzing and Enhancing Darknet Routing // IEEE INFOCOM 2013. — https://www.researchgate.net/publication/261060412_A_Contribution_to_Analyzing_and_Enhancing_Darknet_Routing

20. Roos S., Beck M., Strufe T. VOUTE — Virtual Overlays Using Tree Embeddings. — 2016. — https://arxiv.org/abs/1601.06119 ; see also Socially-Aware Distributed Hash Tables for Decentralized Online Social Networks. — https://arxiv.org/abs/1508.05591

21. Dingledine R., Mathewson N., Syverson P. Tor: The Second-Generation Onion Router // Proc. 13th USENIX Security Symposium, 2004. — https://www.usenix.org/legacy/event/sec04/tech/dingledine.html

22. Jefferys K., Shishmarev M., Harman S. Session: End-To-End Encrypted Conversations With Minimal Metadata Leakage. — 2020. — https://arxiv.org/abs/2002.04609

23. Fall K. A Delay-Tolerant Network Architecture for Challenged Internets // Proc. ACM SIGCOMM 2003. — https://dl.acm.org/doi/10.1145/863955.863960

24. Cerf V. et al. Delay-Tolerant Networking Architecture. — RFC 4838, 2007. — https://www.rfc-editor.org/rfc/rfc4838

25. Burleigh S., Fall K., Birrane E. Bundle Protocol Version 7. — RFC 9171, 2022. — https://www.rfc-editor.org/rfc/rfc9171

26. FidoNet Technical Standards Committee. — http://ftsc.org/

27. Matveev S. NNCP — Node to Node copy. — http://www.nncpgo.org/

28. Qvist M. Reticulum Network Stack: Manual. — https://reticulum.network/ ; https://github.com/markqvist/Reticulum

29. Meshtastic. — https://meshtastic.org/

30. Braem B. et al. A case for research with and on community networks // ACM SIGCOMM Computer Communication Review. — 2013. — Vol. 43, No. 3. — P. 68–73. — https://doi.org/10.1145/2500098.2500108 ; Freifunk: https://freifunk.net/ ; Guifi.net: https://guifi.net/

31. Kleppmann M., Wiggins A., van Hardenberg P., McGranaghan M. Local-first software: you own your data, in spite of the cloud // Onward! 2019. — https://doi.org/10.1145/3359591.3359737 ; https://www.inkandswitch.com/local-first/

32. Tarr D., Lavoie E., Meyer A., Tschudin C. Secure Scuttlebutt: An Identity-Centric Protocol for Subjective and Decentralized Applications // ACM ICN '19. — 2019. — https://doi.org/10.1145/3357150.3357396

33. Nostr: Notes and Other Stuff Transmitted by Relays (NIPs). — https://github.com/nostr-protocol/nips

34. Nevod. Zero Knowledge Infrastructure. — https://nevod.network/

35. Briar Project. — https://briarproject.org/

36. ITU-T Recommendation E.500. Traffic intensity measurement principles. — 1998. — https://www.itu.int/rec/T-REC-E.500

37. ITU-T Recommendation E.412. Network management controls. — 1988. — https://www.itu.int/rec/T-REC-E.412

38. Nagle J. Congestion Control in IP/TCP Internetworks. — RFC 896, 1984. — https://www.rfc-editor.org/rfc/rfc896

39. Chaum D. L. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms // Communications of the ACM. — 1981. — Vol. 24, No. 2. — P. 84–90. — https://doi.org/10.1145/358549.358563

40. Goldschlag D. M., Reed M. G., Syverson P. F. Hiding Routing Information // Information Hiding, LNCS 1174. — Springer, 1996. — P. 137–150. — https://doi.org/10.1007/3-540-61996-8_37

41. Nevod source code. — https://github.com/avk0978/nevod-messenger

Note on method. The review was carried out on open sources; paid databases (IEEE Xplore, ACM Digital Library full texts, Springer, eLibrary) were not searched systematically. Trait scores and classes are expert assessments based on public descriptions of the networks as of September 2026.