Russian Journal of Earth Sciences, Vol. 22, ES6007
https://doi.org/10.2205/2022ES000829

BitTorrent is Apt for Geophysical Data Collection and Distribution

Kirill Kholodkov*[,1] and Igor Aleshin[1,2]

[1] *Schmidt Institute of Physics of the Earth of the Russian Academy of Sciences, Moscow, Russia*
[2] *Geophysical Center of Russian Academy of Sciences, Moscow, Russia*

Received 21 November 2022; accepted 11 December 2022; published 30 December 2022.

This article covers a nouveau idea of a way to collect and handle geophysical data with a peer-to-peer network in near real-time. The text covers a brief introduction to the cause, the technology, and the particular case of collecting data from Global Navigation Satellite System (GNSS) stations. We describe the proof-of-concept implementation that has been tested. The test was conducted with an experimental GNSS station and a data aggregation facility. In the test, original raw GNSS signal measurements were transferred to the data aggregation center and subsequently to the consumer. Our implementation utilized the BitTorrent protocol to communicate and transfer data. The solution could be used to enable the majority of data aggregation centers for fast, reliable, and transparent real-time data handling experience for the benefit of the scientific community.

Keywords: data center, data handling, data acquisition, peer-to-peer network, geophysical observations, GNSS raw, BitTorrent, P2P, GNSS

Citation: Kholodkov, Kirill, and Igor Aleshin, (2022), BitTorrent is Apt for Geophysical Data Collection and Distribution, *Russian Journal of Earth Sciences*, Vol. 22, ES6007, doi: 10.2205/2022ES000829.

1 Introduction

Data plays an important role in geophysics. Among researchers, quality and ease of access are its most valued properties [*Borgman*, 2019]. Whether data quality is the merit of the instrument, the site, or the fieldwork, ease of access is often left out. In many cases fieldwork experiments collect and subsequent work consume data without putting sufficient effort into making sure that the source data would be available to the public, for a reasonable amount of time at least.

We believe this predominantly happens because the scientific community relies on legacy general-purpose technology to publish and distribute data. It is worth noting that some scientific communities created and adopted dedicated technology to meet their data acquisition and handling needs. For instance, the seismological research community developed several [*Havskov et al.*, 2012] protocols and formats to facilitate real-time reliable data acquisition, storage, and processing. It turned out that some of these protocols and formats could be utilized in adjacent scientific fields of expertise – magnetic observation and structural monitoring. That was possible because of the nature of the collected data. Each instrument provides one or several time series streams. Time series is a sequence of numeric values bound to a unique timestamp. The protocols are generally agnostic to the meaning of values that are transferred. This way, we previously adopted [*Ivanov et al.*, 2019] the Seedlink protocol to transfer magnetic and structural monitoring data. However, these solutions only cover the immediate data availability issues and leave data sharing and data reuse problems far behind. When the time comes for accessing data from years ago the way that historical data is stored comes into play. Among different ways [*Saul*, 2014] of getting that data, one can particularly note FTP servers and data ordering with email or automated web form. Both ways feel neither robust nor voguish.

Distributing data across several network sites partially addresses the problem because the data can be found and accessed faster if there is more than one source [*Ahern et al.*, 2014]. To achieve this, data is replicated among several nodes in the network, and special software is utilized to en-

*Corresponding author: keir@ifz.ru

able it. The software is mirroring solutions (rsync-like), distributed filesystem or database, or a peer network. In most cases the use of new and complex software is inevitable. Among the solutions that enable distributed data storage and access we particularly highlight peer-to-peer network data-sharing technologies. These techs have an advantage over others in the task of enabling access to geophysical data – one probably has used them at least once. Indeed, peer networks had great success in delivering data to end users – from updates to operating systems and applications to TV shows. Inherently, we adopted BitTorrent for the needs of geophysical data collection. Below, we will describe both the problem and the solution in detail.

2 State of the Art

Before adopting BitTorrent we explored other data delivery protocols to find a technology combination that is bandwidth-efficient, tolerant to network failures, and has wide software support. For time series, we've found Seedlink to be the optimal solution [*Ivanov et al.*, 2019]. However, neither Seedlink, nor any other time-series protocols and corresponding file formats can be used efficiently to transfer and store complex data – like GNSS measurements or ionograms, and here's why.

Unlike time-series data, these formats contain multiple, in most cases, position-dependent structures. Such structures can be called messages [*GNSS*, 2019], groups [*Galkin et al.*, 2006], packets [*u-Blox*, 2020], etc. Often an instance of a particular structure cannot provide a single complete instrument measurement because it is dependent on either a separate time structure or a similar previous structure, due to differential (or delta) recording. Sometimes, the community offers unified formats, such as RINEX [*Romero*, 2020] to store this data. This kind of unification helps to fight the so-called vendor lock when the user is no longer tied to specific processing software, however, while these solutions combat format diversity they mostly omit transfer issues. With that said we believe it is optimal to conserve the original format for data handling mainly because format unification does not benefit data transfer.

Many data-archiving efforts like those in the IGS [*Infrastructure Committee Central Bureau*, 2015], INTERMAGNET [*Observatories Subcommittee*, 2022], and weather agencies generally around the globe utilize antiquated FTP and SMTP, the one used for email, to transfer data from stations and among datacenters. These protocols are badly inefficient in terms of bandwidth, accessibility (see FTP 'active' mode), and error handling but have vast software support. While FTP shows good transfer speeds in theory, real-life instances are often oversubscribed and limit both speed and number of connections. SMTP adds 30% on average to the payload, and, even if used with private servers, is prone to delays. Both don't provide any error detection and correction relying on TCP and the link level, Ethernet or LTE, to handle this task. However, the undetected error rate is quite observable [*Stone and Partridge*, 2000]. To combat these issues we suggest using peer data-sharing networks. Their protocols are designed to handle large bandwidth, simultaneous delivery, and consistency control on one hand. The drawback, on the other hand, is the lack of production-ready software support for a data acquisition site. In this paper, we describe our experience of adopting a peer network protocol for geophysical data collection and transfer.

As part of the effort on establishing a data aggregation center authors had to perform various data collection incentives: tiltmeter installations, seismic stations, magnetic observatories, GNSS surveys, and so forth. Tiltmeters, seismic stations, and magnetic observatories yield time-series data, so we utilize Seedlink and MQTT to establish real-time data acquisition. In contrast, GNSS measurements do not have a well-known guaranteed data transfer solution like Seedlink. From the station's perspective, the measurements can be stored as bulky files and there should be a way to transfer them reliably to a data center and consumers. The size of files from a single station can reach several gigabytes a day. The size poses a problem for both remote stations'network access channels and distribution centers. From [*Aleshin et al.*, 2014] we also learned that stable connection to the Internet is rather uncommon at remote locations causing data consistency and availability problems. As shown above, we did not feel comfortable with FTP or SMTP so we decided to find another solution to transfer that amount of data. That solution was to use the BitTorrent peer network.

3 BitTorrent

Among several maintained peer-to-peer file sharing network implementations [*Cohen*, 2008; *Klingberg and Manfredi*, 2002; *Kulbak and Bickson*, 2005; *Maymounkov and Mazières*, 2002] we have chosen the one with greater software support and overall popularity [*Bhatia and Rai*, 2017]. At a glance, all these networks share some principles and technology. First, all implementations assume that every participant of the network can transfer data in any direction. Second, every implementation performs a consistency check of the data by calculating and comparing hash values of transferred pieces of data. These hash values and other

knowledge about the data being transferred are called metadata.

However, the way this metadata is distributed and discovered varies in different peer network implementations. BitTorrent, the technology we've chosen, features several ways to do this. The metadata can be written to a file, called torrent-file, and transferred to the peer by other means, including offline ones. Or, the metadata can be formatted as an URL with a protocol called "magnet" and published on the web or transferred by other means. Above this, there's "broadcatching". This technology automates metadata distribution and cuts the required time for distribution to a minimum. Also, BitTorrent has sufficient open software support that allows us to use "broadcatching" as a near-real-time data transfer mechanism. In general, BitTorrent software performs several tasks to enable data sharing.

First, it prepares the metadata of data to be shared. Besides the name, description, and other fields, the metadata includes a hash value calculated for the whole span of data called "infohash" in BitTorrent and hash values for each piece of the data.

Second, the software maintains a list of peers. This list is populated in several ways: using special software called the Torrent tracker, discovering peers on the local network with broadcast or with multicast, participating in a distributed hash table, and manually. Torrent tracker is a separate piece of software that collects infohashes and peer addresses. When a peer queries the tracker on a particular infohash it responds with a list of peers that might have had the data that suits the provided infohash. Local peer discovery allows the discovery of network stations that run BitTorrent software by broadcasting specific packets on the local network and specific multicast addresses. It is described in BitTorrent specification [*Perkins*, 2008; *The 8472*, 2015]. Distributed Hash Table is a software system that runs on a number of nodes on the network and provides lookup services. Almost any BitTorrent peer participates in DHT and publishes known infohashes to it. Other peers may perform a lookup for the specific infohash and obtain a list of peers, just like tracker software does. Manual mode allows to force the software to connect to the specified address and can be used in conjunction with other peer discovery software or catalog, or for testing.

Third, the BitTorrent software performs the transmission of data between peers. When new infohash is added to the running instance the software asks known peers whether they have data that suits the infohash. If a peer has this data the software establishes a connection to that peer and downloads the rest of the metadata (piece hashes, filenames, etc.) and the data piecewise, with every piece checked with the hash provided in the metadata. When part of the data is downloaded and verified the peer can also serve this part of the data to other peers. These processes are performed simultaneously.

In this paper, we utilize broadcatching or "Torrent RSS Feeds" [*Norberg*, 2012]. It is a feature of BitTorrent software that allows the automatic distribution of new content with the help of RSS feeds. RSS (RDF Site Summary or Rich Site Summary) is a structured representation of essential data formatted in XML. This representation is strictly-formatted so that any software that interprets RSS will do so in the same way and will provide similar results. We utilize this mechanism to feed the BitTorrent software of the data consumer with the infohash of the newly available data. According to [*Zhang et al.*, 2009], this technology offers fast and low-latency data transfer.

4 Proof of Concept

To investigate whether the idea is viable for use, we developed software that implements peer-to-peer data transfer of scientific data. In a word, it catches newly collected data from the GNSS Receiver with the help of [*Aleshin et al.*, 2020], computes infohashes, enables data transmission, and presents the RSS feed to potential peers. The software is built upon libtorrent (`http://libtorrent.org/`), Qt, Mongoose, and is called `btcastd` (BEE-TEE-cast-DEE). The three major units comprise `btcastd`: filesystem monitoring and event handling, BitTorrent operations, and on-demand RSS-feed generation and serving, see Figure 1.

The filesystem monitoring portion detects changes to the specified file directory and adds information on new files to the BitTorrent part when these new files are no longer written to. The latter is essential because the infohash of the file is content-dependent. BitTorrent does not allow the infohash to change over time. This way, only finished immutable files are allowed into the BitTorrent portion of the software. To achieve this sort of picking mechanism we relied on particular features of the data collection software. Based on libGREIS the GNSS receiver software has a deep view of the data coming in. Knowing the exact time of data chunks it spreads data sequentially into small files. Files are named after the beginning of the N-minute period. Thus, when the current time passes over to the next N-minute file, the current one is no longer written to, hence it could be added to the BitTorrent portion. Here, we reference this method as "time-dependent". We leverage the filesystem change notification features of Qt to learn about new files. There are

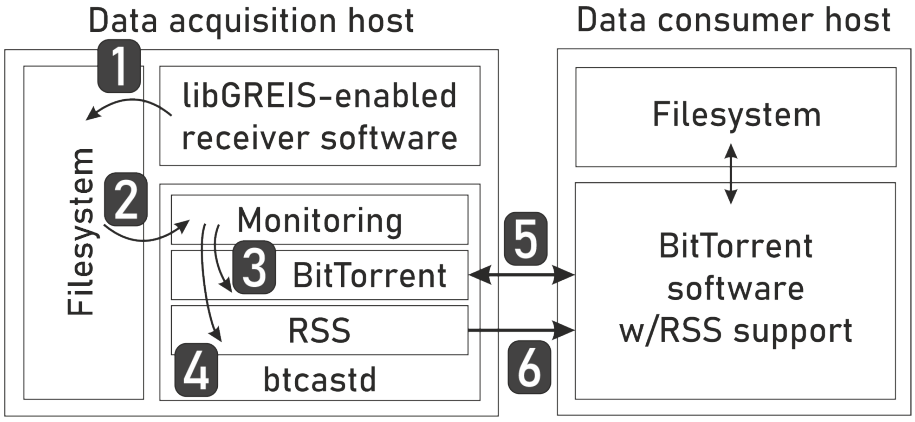

Figure 1: The diagram of the software composition on the data acquisition host and data consumer host. The data consumer host is generally a typical BitTorrent client, while data acquisition host has special software designed for that role. Running at least two separate processes, the data acquisition software, and `btcastd`, the whole stack performs the following tasks (numbers correspond to those on the figure): 1. Acquired data is written to the filesystem; 2. The data is read by monitoring part of `btcastd`; 3. Data is fed to BitTorrent part; 4. Data is fed to RSS part; 5. BitTorrent data transfer operation takes place; 6. RSS is read by BitTorrent software on a data consumer host. The numbers do not depict the order of operation.

many other ways of picking files including passage of notifications from data collecting software, monitoring of low-level file handles, and exercising other methods of such monitoring. However, we've chosen the time-dependent method for the sake of simplicity.

The BitTorrent part performs all actions required to make the file available to other BitTorrent peers. It handles infohash computation, negotiations with peers, data transfer, and managing all network operations except for RSS feed. Most operations are handled by libTorrent.

RSS feed is provided by an integrated HTTP server powered by Mongoose library. The list of infohashes is updated whenever the infohash is added to the BitTorrent portion of the software. The RSS component just serves a specially crafted RSS feed to potential clients.

The software was designed to work in conjunction with GREIS-enabled GNSS receivers through a data-collecting software based on libGreis. Data is collected from receiving hardware and put into files according to a scheme. The scheme allocates file names with the number of minutes since the beginning of the day. These files are put into folders labeled with the day of the year. Next, the days are organized into years. Years comprise the topmost folder in this hierarchy. `Btcastd` monitors these folders and picks all finished files while maintaining the hierarchy path. To save this hierarchy, the path is recorded in the filename portion of the metadata. The resulting RSS feed can be processed with generic BitTorrent software that is RSS-enabled. During the tests we used qBittorrent. qBittorrent was set up to monitor the address of the host where `btcastd` had been running. When a new part of data appeared on the RSS feed qBittorrent started to download this file. We relied on local peer discovery for `btcastd` and qBittorrent to locate each other, although in large-scale deployments use of several torrent trackers is advised.

At some point, the data acquisition host has to delete old data to make storage space available for new measurements. When this happens the corresponding torrents are removed from the RSS feed, thus are no longer available from the original location. Only the relay host and the data center still possess these torrents and associated data. Furthermore, the data center could also create a sort of filing from several pieces of data by generating a new torrent with all those files in it. This will not produce additional copies of data because the hash values of the files still match between the original torrent with only one file and the larger one, but will make it better organized and even more accessible.

5 Conclusion

Presented software could be used to build a sophisticated data handling infrastructure. Within this infrastructure data acquisition hosts (e.g., stations) can reside in remote locations with a suboptimal internet connection and still provide data to the data center or to one of the relay hosts that run BitTorrent software. To minimize the load on the data acquisition hosts the solution can implement whitelisting of peers on the data acquisition host side that prevents any connections except for relays, data centers, and authorized consumer sta-

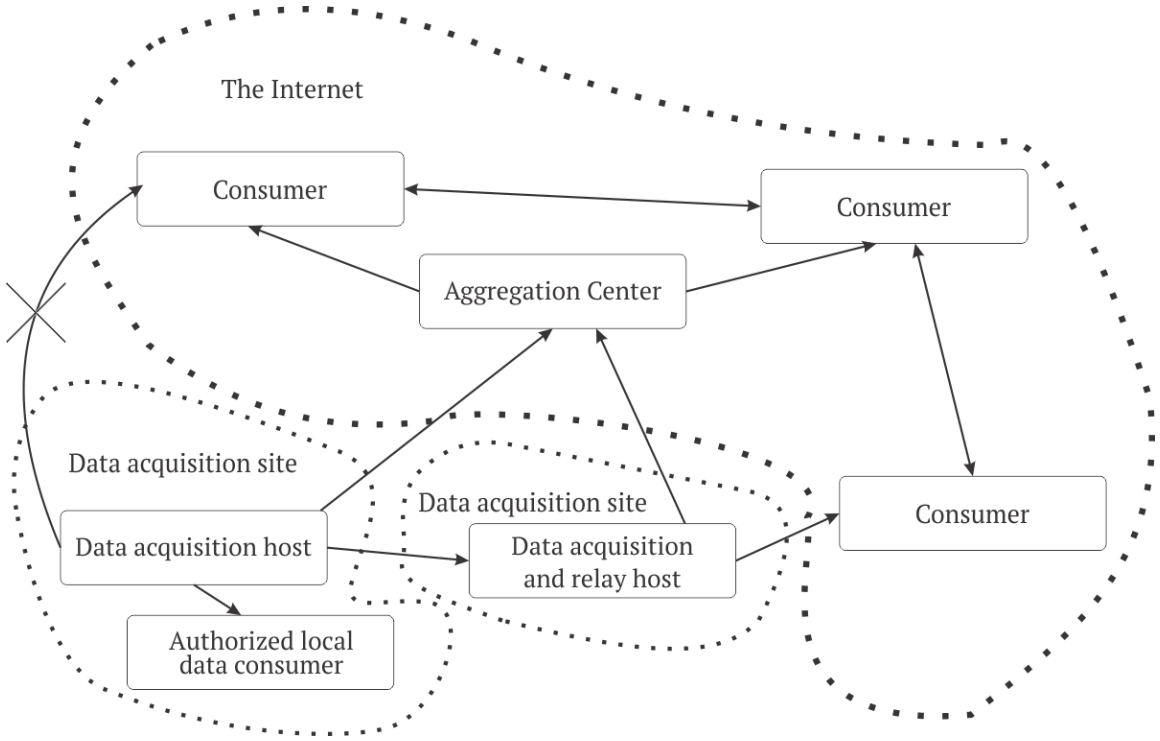

Figure 2: Diagram of potential data flows. Data flow is chained from the data acquisition host to the consumers via the data center and optional relays. Note the possible local connection to the authorized local consumer. Blacklisted connections are marked with X. These connections could be intentionally disabled e.g., to prevent bandwidth abuse at the data acquisition site, although this is optional.

tions e.g., a local watch office can still source data directly from the data acquisition host. Relays run BitTorrent software and collect data from one or several stations to make it available for the data centers, or if the relay has plenty of bandwidth to serve data to consumers on the internet. Due to the nature of peer-to-peer networks, consumers will get the data at a high speed from several locations including other consumers. The diagram is shown in Figure 2.

6 Operational Perspective

This solution has been tested with IPE RAS Data Aggregation Center and an experimental GNSS station located in Moscow, Russia. We used libGREIS-enabled GNSS recording software with Javad Alpha 2 GNSS receiver running on a Raspberry Pi 3 board. Acquired files were processed by `btcastd` and a non-graphical instance of qBittorrent was set at the Center to get all that data with BitTorrent. Consumers were able to collect data similarly. After weeks of operation, the proof-of-concept was deemed viable.

The software implementation is not complex, most of the software could be used out-of-box, e.g., qBittorrent fits well for the consumer role. LibTorrent is simple enough to be used for basic BitTorrent operations, however, things can get quite complicated when it comes to fine-tuning or using advanced features of the library, such as memory and storage I/O management. Diving into these parameters would allow us to combat performance and resource utilization issues to allow smooth operation even on low-power system-on-chip computers although advanced usage is outside the coverage of this article.

It is worth noting that despite the fact that the specification declares the specific way to construct the RSS many BitTorrent implementations construct and parse it in slightly different ways and users may encounter difficulty using broadcatching (e.g., RSS Issues of qBitTorrent on GitHub) until their or counterpart software is updated with bugfix. However, that's again a no-stopper because the majority of BitTorrent software is an open source one can fix or workaround the issue *in situ*.

We hope to further develop the idea and software implementation to enrich the functionality of applicable data acquisition activities.

The source is available at https://github.com/iperas/omnicollect

Acknowledgments. The work was carried out within the framework of the state assignment of the Schmidt Institute of Physics of the Earth of the Russian Academy of Sciences (IPE RAS) and

Geophysical Center of the Russian Academy of Sciences (GC RAS), approved by the Ministry of Education and Science of the Russian Federation. The collaboration between the institutes was enabled by the separate agreement on joint scientific activities between the IPE RAS and the GC RAS.

References

Ahern, T., R. Benson, R. Casey, C. Trabant, and B. Weertman (2014), Improvements in Data Quality Integration and Reliability: New Developments at the IRIS DMC, *Advances in Geosciences*, *40*, 31–35, doi:10.5194/adgeo-40-31-2015.

Aleshin, I., S. Burguchev, K. Kholodkov, V. Alpatov, and A. Vasiliev (2014), Data Handling in GNSS Receiver Network and Ionosphere Monitoring Service Solution, in *2014 International Conference on Engineering and Telecommunication*, pp. 122–125, doi:10.1109/EnT.2014.32.

Aleshin, I. M., K. I. Kholodkov, and V. N. Koryagin (2020), Framework for GREIS-formatted GNSS data manipulation, *GPS Solutions*, *24*(2), 63, doi:10.1007/s10291-020-0971-7.

Bhatia, M., and M. K. Rai (2017), Identifying P2P traffic: A survey, *Peer-to-Peer Networking and Applications*, *10*(5), 1182–1203, doi:10.1007/s12083-016-0471-2.

Borgman, C. L. (2019), The Lives and After Lives of Data, *Harvard Data Science Review*, *1*(1), doi:10.1162/99608f92.9a36bdb6.

Cohen, B. (2008), The BitTorrent Protocol Specification, http://bittorrent.org/beps/bep_0003.html.

Galkin, I. A., B. W. Reinisch, and R. Gamache (2006), *Standard Archiving Output (SAO) Format*, University of Massachusetts Lowell.

GNSS (2019), Javad GNSS Inc. and GNSS Receiver External Interface Specification, Revision 3.7.6 Edition.

Havskov, J., L. Ottemöller, A. Trnkoczy, and P. Bormann (2012), Seismic Networks, *New Manual of Seismological Observatory Practice 2 (NMSOP2)*, doi:10.2312/GFZ.NMSOP-2_CH8.

Infrastructure Committee Central Bureau (2015), IGS Site Guidelines.

Ivanov, S. D., I. M. Aleshin, V. N. Koryagin, F. V. Perederin, and K. I. Kholodkov (2019), Geophysical data aggregation center IPE RAS, CEUR Workshop Proceedings, in *Proceedings of the V International Conference Information Technologies in Earth Sciences and Applications for Geology, Mining and Economy (ITES&MP-2019) Moscow, Russia, October 14–18*, pp. 52–56.

Klingberg, T., and R. Manfredi (2002), *Gnutella 0.6*, Gnutella Protocol Development.

Kulbak, Y., and D. Bickson (2005), *The eMule Protocol Specification*, The Hebrew University of Jerusalem, Jerusalem.

Maymounkov, P., and D. Mazières (2002), Kademlia: A Peer-to-Peer Information System Based on the XOR Metric, in *Peer-to-Peer Systems*, edited by P. Druschel, F. Kaashoek, and A. Rowstron, pp. 53–65, Springer Berlin Heidelberg, Berlin, Heidelberg.

Norberg, A. (2012), Torrent RSS Feeds, http://bittorrent.org/beps/bep_0036.html.

Observatories Subcommittee (2022), INTERMAGNET Observatory Application Form, https://intermagnet.github.io/membership.html, Acessed: November 2022.

Perkins, R. (2008), Zeroconf Peer Advertising and Discovery, http://bittorrent.org/beps/bep_0026.html.

Romero, I. (Ed.) (2020), *The Receiver Independent Exchange Format*, ESA/ESOC/Navigation Support Office, Darmstadt, Germany.

Saul, J. (2014), Seismic waveform data retrieval, in *New Manual of Seismological Observatory Practice 2 (NMSOP-2)*, Deutsches GeoForschungsZentrum Gfz, doi:10.2312/GFZ.NMSOP-2_IS_8.3.

Stone, J., and C. Partridge (2000), When the CRC and TCP Checksum Disagree, in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, SIGCOMM '00*, pp. 309–319, Association for Computing Machinery, New York, NY, USA, doi:10.1145/347059.347561.

The 8472 (2015), Local Service Discovery, http://bittorrent.org/beps/bep_0014.html.

u-Blox (2020), u-Blox AG, F9 High Precision GNSS Receiver Interface Description.

Zhang, Z., Y. Lin, Y. Chen, Y. Xiong, J. Shen, H. Liu, B. Deng, and X. Li (2009), Experimental Study of Broadcatching in BitTorrent, in *2009 6th IEEE Consumer Communications and Networking Conference*, pp. 1–5, doi:10.1109/CCNC.2009.4784862.