Low-Latency Networking: Architecture, Techniques, and Opportunities

Xutong Zuo,
Tsinghua University

Yong Cui
Tsinghua University

Mowei Wang
Tsinghua University

Tianxiao Xiao
Syracuse University

Xin Wang
Stony Brook University

Editor:
Yong Cui
cuiyong@tsinghua.edu.cn

With the advent of delay-sensitive applications, low-latency networking is attracting research attention from academia, industry, and standards organizations. This article analyzes the causes of latency across network architecture, reviews some state-of-the-art techniques to reduce latency, and presents several opportunities.

The emergence of new applications and operational scenarios places exacting requirements on latency. For example, high user-perceived latency in a cloud game deteriorates players' interactions and degrades the user experience. In industrial operations, control systems depend on low network latency, which is often required to be within several to hundreds of milliseconds to achieve real-time control.[1]

However, the Internet was designed originally to provide best-effort delivery and does not guarantee any quality of service (latency included). In addition, network latency is impacted by many factors such as routing decisions and network traffic. As a result, achieving low network latency is a challenging but critical problem that has attracted great attention. For instance, the former chair of the Internet Engineering Task Force (IETF) discussed design choices of applications that require low latency from a system perspective.[2] Moreover, a broad survey was organized to classify techniques by latency sources, which provided a comprehensive understanding of the root causes of latency.[3]

Unlike previous taxonomies, we analyze latency and techniques by following layers of the network architecture, which naturally integrates with network standards. We attempt to present the issue of latency from an architectural perspective, and in doing so, we hope to facilitate the development of IETF standards.

Both the requirements and the sources of latency vary with different functionalities on different layers. Utilizing services provided by lower layers, upper layers with more functionalities will introduce additional delay. On the application layer, many applications focus on the completion time of transmitting a data block that might consist of several packets. On the transport layer, if reliable and ordered transmission is provided, retransmission delay and head-of-line (HOL) blocking are added on the basis of the round-trip time (RTT).

Lower layers focus on the latency of delivering every packet. Routing on the network layer determines the paths, leading to queuing delay. The link layer is responsible for transferring datagrams between adjacent network nodes, where latency consists of channel access delay due to the shared medium.

Here, we present a brief survey of network latency and approaches to reduce latency, particularly the delays resulting from the protocol design and functionalities. We first summarize factors impacting delay from different layers of the network architecture. Then, we review some state-of-the-art solutions to reduce latency at each layer. Finally, opportunities for future work and concluding remarks are given.

LATENCY ACROSS THE NETWORK ARCHITECTURE

In this section, we analyze the main causes of latency at each layer of the TCP/IP architecture. As Figure 1 shows, each layer involves various functions that trigger latency, such as congestion control on the transport layer and routing on the network layer. To implement functions, related protocols have been designed, and mechanisms in these protocols introduce delays, such as TCP handshake delay or loss recovery delay. Note that latency and delay are used interchangeably in this article.

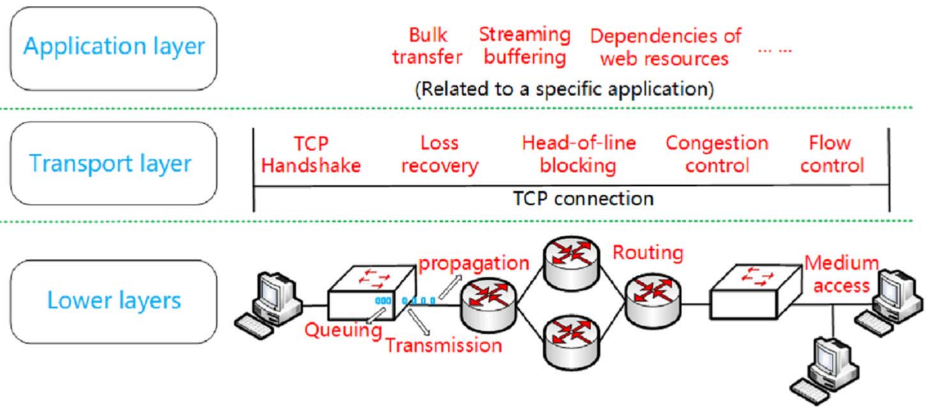

Figure 1. Latency at each layer of the TCP/IP architecture. Different causes of latency and factors affecting latency are shown according to the network architecture.

Latency Specific to Applications

Applications with disparate design goals can have different sources of latency. One of the most important concerns of latency is the block completion time. Only when all units in the block (e.g., chunks of file transfers and frames in video streaming) are transferred to the receiver can the data be used by the application. In the following text, we take two typical applications as examples to introduce other sources of delay.

Interactive live streaming demands low communication latency, which is different from the traditional video-on-demand application. Viewers who comment on broadcasts need immediate feedback; the Real-Time Messaging Protocol (RTMP) is applied to help achieve low latency. The end-to-end delay from broadcasters to viewers adopting RTMP includes upload delay (the delay for transferring a frame from the broadcaster to the server), last-mile delay (the delay for transferring a frame from the server to the viewer), and client-buffering delay (the gap between the frame arrival time and frame playing time).[4] The client buffer aims to prevent playout interruption, but a long buffering delay on the application layer is introduced, which accounts for the largest proportion of the end-to-end latency.

For web services, the page-loading time can be defined as the duration from the receipt of the user request to the display of the whole page. Measurements and analyses have shown that latency plays a defining role concerning page-loading time as well as bandwidth.[5] Factors that contribute to high latency on the application layer include the TLS/SSL handshake (required for secure connections) delay, HTTP redirections (redirecting a client request to a different location), HTTP blocking (the waiting time caused by the maximum number of TCP connections to a server in a browser), and dependencies between web resources (evaluating previous objects and fetching new resources).[6]

Latency of Data Transmission on Lower Layers

The latency on the transport layer is determined by the transport mechanism. Reliable, ordered transmission incurs high latency; we focus on per-connection latency in this case. The per-connection latency comprises the protocol handshake, which creates a connection setup delay that bears on the overall transmission delay, especially for short flows. Moreover, to ensure reliable and ordered delivery of a series of packets, the retrieval and retransmission of a lost segment incur a tremendous delay, which also produces HOL blocking by holding up subsequent segments.

Routing is a core functionality of the network layer. Unlike the other sources of delay presented in this article, the latency of the routing operation generally has been neglected, but routing decisions greatly impact latency. A selected path with more hops might introduce higher packet processing and propagation delay as packets traverse more network devices. Besides, the queuing delay fluctuates significantly with the routes, leading to varying RTTs.

For a shared medium, media access control (MAC) on the link layer addresses the shared-channel-allocation problem. Additional latency arises from the design of MAC protocols. Static channel allocation, such as time division multiple access (TDMA), gives rise to predictable latency due to the fixed assignment of the shared medium. For TDMA, the sender waits for its assigned time slot, rendering the unused time slot idle and thus increasing the waiting time, especially when the channel load is low. As for dynamic allocation, the collision and buffering delay resulting from the contention channel cannot be ignored, especially when the channel load is high. Channel competition makes the transmission delay of a frame erratic, which is not suitable for real-time traffic.

The queuing delay at each device contributes a large part to the end-to-end latency, which is caused by the contention of shared sources, such as output ports and processing units. The buffer is set to hold queued packets, preventing packet loss and handling network burst traffic, which incurs a queuing delay. Many factors affect the queuing delay, including routing decisions, congestion control, etc. For longer links, such as satellite links, propagation delay is nontrivial on the lower layers.

NOVEL TECHNIQUES TO REDUCE LATENCY

To achieve low network latency, we can consider the delay created at each layer of the TCP/IP network architecture. In this section, we discuss some research efforts dedicated to reducing the latency mentioned above.

Reducing Application-Specific Latency

First, we introduce two typical applications whose application layer latency we discussed earlier: web services and interactive live streaming. Then, we present the corresponding methods on the application layer for reducing the latency.

For web services using HTTP, unnecessary handshakes incur latency during connection setup. Many procedures performed on the application layer improve the handshake delay by reducing the number of TCP connections. HTTP 1.1 adopts persistent connections to deal with multiple HTTP requests in one TCP connection.[7] However, the support for concurrent connections brings latency and gives rise to mounting complexity of management.

Fortunately, the multiplexing adopted by HTTP 2.0[8] addresses this problem. Several requests or responses sent to the same receiver are multiplexed while being transferred. Labels are applied to distinguish between different streams and different packets. This eliminates the need for establishing multiple TCP connections and contributes to the reduction of page-loading time. In fact, HTTP 2.0 has received the attention of the IETF and was standardized by the HTTPBIS (Hypertext Transfer Protocol) Working Group.

For live streaming, smooth playout and low latency are conducive to a good user experience. To achieve continuous playout when the condition of the best-effort network varies, buffering on the client side is employed. However, an inevitable delay comes with the buffer. Measurements show that the buffering latency is the largest part of the live streaming delay.[6]

To reduce buffering latency without suffering from stalls, the amount of prebuffered data and the adjustment of the playout rate are worthy of study. For example, adaptive media playout is an effective methodology that aims at playout rate adjustment and has proven highly successful in buffer latency reduction. Specifically, an attempt has been made to leverage historical buffer level variation information and estimate the buffer variation range, after which buffer size is adjusted accordingly.[9] This method reduces the latency under good conditions, using a shallow buffer to guarantee no interruption.

Optimizing the TCP Handshake and Alleviating HOL Blocking

For a single TCP connection, the traditional three-way handshake adds latency to data transmission, especially for short flows. This latency can be mitigated by reducing the number of control interactions in a single TCP connection. TCP Fast Open (TFO) aims to start transmitting data while carrying on the handshake.[10] It is achieved by using a cookie and sending data to the receiver before an acknowledgment arrives. If a TLS handshake is required to provide a secure connection, one to two more RTTs are introduced on top of conventional TCP.

Quick UDP Internet Connections (QUIC)[11] can achieve one RTT handshake for the first-time connection by incorporating the transport and crypto handshake. For subsequent connections, clients send the cached cryptographic cookie, the encrypted payload, and other information to the server, which the server can utilize to authenticate clients and decrypt the payload data. As a result, a zero-RTT handshake is attained. Internet statistics show that QUIC was supported by about 1.0 percent of all websites as of August 2018.[12]

Another delay contributor on the transport layer is HOL blocking, which is caused by lost or out-of-order packets in the sequential packet delivery. Methods that speed up the retransmission of lost packets help relieve HOL blocking. Keeping the sender aware of the packet loss in a timely manner, instead of waiting for the retransmission timeout, can accelerate the retransmission. In fast retransmit,[13] three (or fewer in early retransmit[14]) duplicate ACKs indicate packet loss and trigger the retransmission. An explicit notification of packet loss can also notify the sender to retransmit with dispatch. The cutting-payload mechanism trims the packet, leaves the header transferred to the receiver, and sends negative acknowledgments to inform the sender of the packet loss.[15] By decreasing the time to retransmit the lost packet, the waiting time of subsequent packets is reduced.

Some new protocols also have countermeasures upon HOL blocking. For instance, QUIC deals with this problem by supporting multiple streams in one connection. To be more specific, one QUIC packet consists of several frames of a small number of streams. Thus, a lost packet only causes HOL blocking of the corresponding streams, rather than all streams.

Also, in Multipath TCP, because the packets take different paths with different RTTs, they might arrive at the receiver out of order. To solve this problem, opportunistic retransmission has been proposed, in which the message causing the HOL blocking is re-sent on subflows that have available congestion windows.[16]

Moreover, another method, Slide Together Multipath Scheduler, allows preallocating packets for the fast path and allows packets with a larger sequence number to travel on slow paths.[17] In this way, packets can arrive at the receiver in order without HOL blocking; more details can be seen in Figure 2.

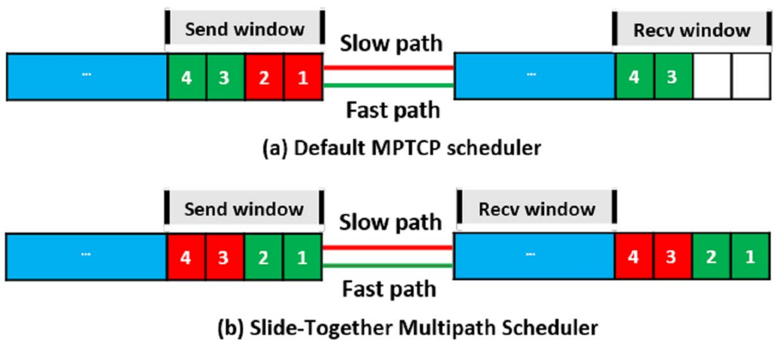

Figure 2. The Slide Together Multipath Scheduler (STMS) prevents HOL (head-of-line) blocking. If the fast path is not available when the sender starts sending data, the default Multipath TCP (MPTCP) scheduler will send packets with small numbers on the slow path, which causes HOL blocking. However, when STMS is adopted, the slow path always sends packets with sequence numbers larger than those of the fast path.

Reducing Latency on the Lower Layers

Routing is a fundamental functionality on the network layer that affects RTT by selecting paths. However, existing routing protocols were designed without considering latency. Methods based on traffic engineering can be used to reduce latency, but they are often limited by inadequate models. Queuing theory is a common way to model network queuing and assist traffic routing. Nonetheless, it might not perform well because in most cases it is a queued network rather than a single queue that is dealt with. A model-free approach utilizes deep reinforcement learning, bypassing the problem of building an accurate mathematical model.[18] Despite the potentially high performance, it might face some challenges; e.g., it is hard to obtain the required real-time reward.

Some companies, such as ViewQwest and Amazon, are also studying latency-based routing to reduce RTT and meet the requirement of delay-sensitive applications. For example, ViewQwest benefits from redundancy by monitoring traffic and probing all available paths to select the best route for low latency.[19]

Frame aggregation is an effective enhancement to improve throughput and reduce medium access delay on the link layer. It reduces the transmission delay and buffering delay by abating the overhead of the frame header and the number of contentions, respectively. However, a tradeoff exists between the time spent in computing aggregates and the time saved owing to aggregating frames. Operation with high efficiency might be complicated and consume much more time.[20] In addition, a larger aggregated frame means a longer waiting time before channel access. Existing aggregation methods mostly aim at improving throughput and transmission efficiency. Reducing latency while optimizing the above two performance metrics is still an open issue. Frame size adaptation considering channel conditions might also be a choice.

Reducing Queuing Delay

Queuing delay is one of the most prominent factors that contribute to high end-to-end latency, especially for datacenters with small internal distances. A priority mechanism can reduce flow completion time for delay-sensitive flows.[21] To prevent the blockage of delay-sensitive flows, their priority can be raised with the use of a priority queue scheduler.

A well-designed congestion control algorithm can also help prevent queues from accumulating, thus directly reducing queuing delay. A new delay-based congestion control algorithm, Copa, has been proposed to achieve low queuing delay and high throughput.[22] It sets an objective function with packet delay as a penalty, and the target sending rate is proportional to the reciprocal of the queue delay. If the current rate exceeds the target rate, the sender reduces the congestion window, which prevents the accumulation of queues.

Google Congestion Control,[23] an architecture proposed for Web Real-Time Communication (WebRTC), has a delay-based controller to directly manage the rate of the sender. The (one way) queuing delay gradient (the derivative of the queuing delay) serves as the signal to minimize the queuing delay along the end-to-end path. The derivative can reflect the change of the buffer, which provides prescience of buffer size and can be leveraged to reduce latency. Kalman filtering is designed to estimate the queuing delay gradient, and an adaptive threshold of this gradient is set to control the rate of increase or decrease, which helps to minimize the buffer size and queuing delay.

OPPORTUNITIES

Novel applications and scenarios have sprung up that place high requirements regarding latency. In addition, emerging techniques, such as edge computing and data-driven methods, are being applied to reduce latency in many scenarios.

New Applications and Scenarios

An ocean of emerging applications that require ultralow latency have come into existence. One example is VR, which needs low motion-to-photon latency to create the sense of reality. For VR, latency consists of computation time and network transfer time, and the tradeoff between them is under research.

For example, on the basis of the knowledge that VR content rendering can be divided into foreground interactions and predictable background virtual environments, phone–server cooperative rendering has been proposed and proves promising.[24] Foreground rendering is completed at the mobile local GPU. Background prerendering and prefetching are carried out on the server. Rendering the interaction locally can provide a better interaction experience and avoid a long transmission time in the network.

Datacenters are a representative network scenario of low latency. Unlike traditional networks, inside datacenters, architecture and protocols can be designed and modified flexibly for high performance. For instance, existing wireless datacenters face some challenges—e.g. dense interfaces and limited wireless links. A redesign of wireless datacenters using a multireflection ring topology has been proposed.[25] Leveraging a flat reflector, the wireless signal can be reflected several times and transmitted to the target server without traversing multiple hops. This circumvents the queuing and processing delay in intermediate devices.

Furthermore, remote direct memory access (RDMA) supports zero-copy technology and can complete the data transmission without occupying the CPU, permitting low-latency networking. This eliminates the bottleneck of datacenter networks.

Novel Methods to Reduce Latency

There are many novel methods for reducing latency. Fog computing and edge computing put content or services near the user to help reduce physical distance and provide “local computation” capabilities. Delay-sensitive management systems of applications in response to changes of network latency can facilitate the intelligent distribution of content to reduce latency. What’s more, using Internet of Things gateways or local processors as the main computing devices for applications avoids the time to pass all information back and forth from the central remote datacenter.

With the development of machine-learning techniques, data-driven methods can facilitate networking optimization and control, including bitrate adaptation, congestion control, and traffic engineering.[26] They can also help achieve low latency. For example, deep learning can be leveraged to solve the topology adaptation problem in datacenter networks, with traffic demand as the input and a near-optimal topology as the output.[27] With an expressive learning framework, this method can flexibly support optimization over both flow-level or application-level objective functions, including the demand completion time or Hadoop job completion time.

CONCLUSION

Low-latency networking is worth studying and is of critical importance for emerging applications such as VR. This article has presented a short survey on the causes of latency at each layer of the TCP/IP network architecture and different techniques to achieve low latency. We hope that the analysis in this article helps drive the effort forward.

In addition, low-latency networking can enable the development of new infrastructure and new methods. However, challenges and opportunities coexist. We thus encourage the continuous and in-depth study of this problem, which requires combining the competencies of academia and industry.

ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China project 61872211 and the National Key R&D Program of China (2017YFB1010002). Xin Wang's research is supported by the US National Science Foundation Computer and Network Systems award 1526843.

REFERENCES

1. *Industrial Routing Requirements in Low-Power and Lossy Networks*, K. Pister et al., IETF RFC RFC 5673, IETF, 2009; https://tools.ietf.org/html/rfc5673.
2. *Low Latency Applications and the Internet Architecture*, J. Arkko et al., IETF draft draft-arkko-arch-low-latency-02, IETF, 2017; https://tools.ietf.org/html/draft-arkko-arch-low-latency-02.
3. B. Briscoe et al., "Reducing Internet Latency: A Survey of Techniques and their Merits," *IEEE Comms. Surveys & Tutorials*, vol. 18, no. 3, 2016, pp. 2149–2196.
4. B. Wang et al., "Anatomy of a Personalized Live Streaming System," *Proceedings of the 2016 Internet Measurement Conference* (IMC 16), 2016, pp. 485–498.
5. M Belshe, "More Bandwidth Doesn't Matter (Much)," Google, 8 April 2010; https://goo.gl/PFDGMi.
6. R. Netravali et al., "Polaris: Faster Page Loads Using Fine-grained Dependency Tracking," *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation* (NSDI 16), 2016, pp. 123–136.
7. R. Fielding et al., *Hypertext Transfer Protocol—HTTP/1.1*, R. Fielding et al., IETF RFC RFC 2616, IETF, 1999; https://tools.ietf.org/html/rfc2616.
8. *Hypertext Transfer Protocol Version 2 (HTTP/2)*, M. Belshe et al., IETF RFC RFC 7540, IETF, 2015; https://tools.ietf.org/html/rfc7540.
9. J. Wang et al., "Adaptive media playout buffer management for latency optimization of mobile live streaming," *2017 IEEE International Conference on Multimedia & Expo Workshops* (ICMEW 17), 2017, pp. 369–374.
10. *TCP Fast Open*, Y. Cheng et al., IETF RFC RFC 7413, IETF, 2014; https://tools.ietf.org/html/rfc7413.
11. Y. Cui, "Innovating transport with QUIC: Design approaches and research challenges," *IEEE Internet Computing*, vol. 21, no. 2, 2017, pp. 72–76.
12. "Usage of Site Elements for Websites," Web Technology Surveys; https://w3techs.com/technologies/overview/site_element/all.
13. *TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms*, W. Stevens et al., IETF RFC RFC 2001, IETF, 1997; https://tools.ietf.org/html/rfc2001.
14. *Early Retransmit for TCP and Stream Control Transmission Protocol (SCTP)*, M. Allman et al., IETF RFC RFC 5827, IETF, 2010; https://tools.ietf.org/html/rfc5827.

15. P. Cheng et al., "Catch the whole lot in an action: Rapid precise packet loss notification in data centers," *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation* (NSDI 14), 2014, pp. 17–28.
16. C. Paasch et al., "Experimental evaluation of multipath TCP schedulers," *Proc. 2014 ACM SIGCOMM Workshop on Capacity Sharing* (CSWS 14), 2014, pp. 27–32.
17. H. Shi et al., "STMS: Improving MPTCP Throughput Under Heterogeneous Networks," *USENIX Annual Technical Conference* (ATC 18), 2018, pp. 719–730.
18. Z. Xu et al., "Experience-driven networking: A deep reinforcement learning based approach," *IEEE Infocom*, 2018.
19. ViewQwest Content Hub, blog; http://blog.viewqwest.com/what-is-latency-based-routing.
20. D. Skordoulis et al., "IEEE 802.11 n MAC frame aggregation mechanisms for next-generation high-throughput WLANs," *IEEE Wireless Communications*, vol. 15, no. 1, 2008, pp. 40–47.
21. M. Alizadeh et al., "pFabric: Minimal Near-Optimal Datacenter Transport," *Proceedings of the ACM SIGCOMM 2013 conference* (SIGCOMM 13), 2013; doi.org/10.1145/2486001.2486031.
22. V. Arun et al., "Copa: Practical Delay-Based Congestion Control for the Internet," *15th Usenix Symp. on Networked Systems Design and Implementation*, 2018.
23. G. Carlucci et al., "Analysis and design of the google congestion control for web real-time communication (WebRTC)," *Proceedings of the 7th International Conference on Multimedia Systems*, 2016, p. 13.
24. Z. Lai et al., "Furion: Engineering High-Quality Immersive Virtual Reality on Today's Mobile Devices," *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, 2017.
25. Y. Cui et al., "Diamond: Nesting the data center network with wireless rings in 3-d space," *IEEE/ACM Transactions on Networking*, vol. 26, no. 1, 2018, pp. 145–160.
26. M. Wang et al., "Machine learning for networking: Workflow, advances and opportunities," *IEEE Network*, vol. 32, no. 2, 2018, pp. 92–99.
27. M. Wang et al., "Neural Network Meets DCN: Traffic-driven Topology Adaptation with Deep Learning," *Proceedings of the ACM on Measurement and Analysis of Computing Systems 2.2*, 2018, p. 26.

ABOUT THE AUTHORS

Xutong Zuo is a PhD student in Tsinghua University's Department of Computer Science and Technology. Her research interests include low-latency networking and live streaming. Zuo received a BE in computer science and technology from the Beijing University of Posts and Telecommunications. Contact her at zuoxt18@mails.tsinghua.edu.cn.

Yong Cui is a full professor in Tsinghua University's Department of Computer Science and Technology. His research interests include computer network architecture and mobile computing. Cui received a PhD in computer science from Tsinghua University. He's the corresponding author. Contact him at cuiyong@tsinghua.edu.cn.

Mowei Wang is working toward his PhD in Tsinghua University's Department of Computer Science and Technology. His research interests are in datacenter networks and machine learning. Wang received a BE in communication engineering from the Beijing University of Posts and Telecommunications. Contact him at wang.mowei@outlook.com.

Tianxiao Xiao is an undergraduate student in Syracuse University's Department of Engineering and Computer Science. His research interests are in machine learning and networking. Contact him at williamshawxtx@gmail.com.

Xin Wang is an associate professor in Stony Brook University's Department of Electrical and Computer Engineering. Her research interests include mobile computing and wireless networking. Wang received a PhD in electrical and computer engineering from Columbia University. Contact her at x.wang@stonybrook.edu.

Contact department editor Yong Cui at cuiyong@tsinghua.edu.cn.