LiveNet: A Low-Latency Video Transport Network for Large-Scale Live Streaming

Jinyang Li[†] [§] [‡], Zhenyu Li[†] [‡], Ri Lu[§], Kai Xiao[§], Songlin Li[§], Jufeng Chen[§], Jingyu Yang[§], Chunli Zong[§], Aiyun Chen[§], Qinghua Wu[†] [‡], Chen Sun[§], Gareth Tyson[*], Hongqiang Harry Liu[§]

[†]Institute of Computing Technology, Chinese Academy of Sciences [§]Alibaba Group
[‡]University of Chinese Academy of Sciences [*]Hong Kong University of Science and Technology (GZ)

ABSTRACT

Low-latency live streaming has imposed stringent latency requirements on video transport networks. In this paper, we report on the design and operation of the Alibaba low-latency video transport network, LiveNet. LiveNet builds on a flat CDN overlay with a centralized controller for global optimization. As part of this, we present our design of the global routing computation and path assignment, as well as our fast data transmission architecture with fine-grained control of video frames. The performance results obtained from three years of operation demonstrate the effectiveness of LiveNet in improving CDN performance and QoE metrics. Compared with our prior state-of-the-art hierarchical CDN deployment, LiveNet halves the CDN delay and ensures 98% of views do not experience stalls and that 95% can start playback within 1 second. We further report our experiences of running LiveNet over the last 3 years.

CCS CONCEPTS

• **Networks** → **Overlay and other logical network structures**; *Network control algorithms*;

KEYWORDS

CDN; Low latency transmission; Live streaming

ACM Reference Format:
Jinyang Li[† § ‡], Zhenyu Li[† ‡], Ri Lu[§], Kai Xiao[§], Songlin Li[§], Jufeng Chen[§], Jingyu Yang[§], Chunli Zong[§], Aiyun Chen[§], Qinghua Wu[† ‡], Chen Sun[§], Gareth Tyson[*], Hongqiang Harry Liu[§]. 2022. LiveNet: A Low-Latency Video Transport Network for Large-Scale Live Streaming. In *ACM SIGCOMM 2022 Conference (SIGCOMM '22), August 22–26, 2022, Amsterdam, Netherlands.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3544216.3544236

Co-corresponding authors: Zhenyu Li, Ri Lu.

SIGCOMM '22, August 22–26, 2022, Amsterdam, Netherlands

ACM ISBN 978-1-4503-9420-8/22/08.
https://doi.org/10.1145/3544216.3544236

1 INTRODUCTION

With the global pandemic, live streaming has become a daily necessity. As new Low-Latency Live Streaming use cases emerge (e.g., e-commerce, work, entertainment gaming), the number of users has grown remarkably [2]. This has been accompanied by growing user expectations, challenging the flexibility and scalability of underlying transmission systems.

As one of the world's primary CDN providers, Alibaba's CDN [5] hosts many live streaming applications (e.g., Taobao Live [7] for e-commerce). For several years, these applications have largely been underpinned by Alibaba's first-generation hierarchical video transport network, where the stream is passed to a central system for media processing and then distributed to edge nodes that subsequently connect to the viewers. Often, these CDN nodes form a (multi-layer) overlay tree, with leaf nodes serving clients and internal nodes disseminating content to the leaves, e.g., using application-layer multicast [11] and caching [12, 38, 39]. This, however, places significant pressure on the central processing system and internal nodes, which must scale with the number of streams. This can become particularly challenging for live streams with strict delay constraints due to the need for streams to traverse the depth of the delivery tree twice. Indeed, as we show in §2, the tree-structured overlay falls short of the stringent requirements on CDN delay that is imposed by low-latency live streaming services.

In this paper, we report our work in building and operating Alibaba Cloud's live streaming service, LiveNet. For several years, this has been built upon the hierarchical CDN model described above. Although this system has helped us successfully host several large-scale live events (such as the Double 11 Shopping Festival [6] and the FIFA World Cup [1]), the complexity of scaling a rigid hierarchical model has proven difficult from an operational and cost perspective. For example, we have found that many of our edge (leaf) nodes remain underutilized, while our root nodes are heavily overloaded. Further, we have observed that user demand can vary on a per-second basis, making fine-grained resource allocation important, but challenging without the ability to dynamically restructure the overlay (tree) topology to avoid hot spots. More problematic still, we often have both latency tolerant and intolerant streams following identical paths through our delivery trees, despite their differing needs. In sum, the rigidity of a hierarchical model is no longer sufficient to support our diverse application requirements.

With the above in mind, three years ago we set out to re-architect our live video streaming platform. Here, we present LiveNet — Alibaba's centrally coordinated low-latency video network based on a flat CDN structure. We focus on introducing three main *design choices* that build on our prior operational experience.

First, to move away from rigidly fixed overlay topologies or pre-assigned roles for nodes (§2), LiveNet is built upon a flat CDN similar to [29, 37]. At its core is a set of flexible nodes (each a cluster of machines) that can serve several dynamically allocated roles: *producers* (which receive and process streams from broadcasters), *consumers* (which receive requests from clients and implement fine-control on streams), and *relays* (which interconnect the consumers and producers in an arbitrary overlay topology, offering services such as forwarding and caching). By decoupling the individual nodes from any particular role, it becomes possible to compose the most appropriate overlay topology on a per-application basis and distribute the load evenly, avoiding prior central hot spots.

This flexibility naturally comes with a number of resource allocation and management challenges, particularly when operating at scale. Thus, our **second** design choice borrows from Software Defined Networking (§4): Rather than embedding control logic in each node, we design a logically centralized CDN controller (the *Streaming Brain*) that is responsible for designating roles to nodes, computing overlay paths between them, and selecting paths for consumer nodes. By logically centralizing management functionality, we can easily experiment with new topologies and configurations to bypass problematic nodes or implement per-application policies.

Whereas the above architecture allows us to flexibly compose new overlay topologies with embedded services at each hop (e.g., caching, transcoding), it also introduces challenging overheads. This is because packets must traverse multiple software stacks, introducing undesirable delays for our live streaming customers. To alleviate this, our **third** design choice uses a novel stream forwarding mechanism with the goal of minimizing end-to-end delay (§5). This is based on two parallel packet processing paths within each node – a fast & slow path, implementing different functions provided by different protocol stack layers. In this model, upon receiving a (RTP) packet, each node immediately forwards it to the next hop in the overlay, without performing traditional control functions such as loss detection or congestion control (the *fast path*). In parallel, a copy of the packet is replicated onto the,*slow path*[1] which introduces congestion control and loss recovery in case the fast path experiences a loss, and also implements GoP (Group of Pictures) caching on each node. This optimizes LiveNet for delay — the fast path delivers packets as quickly as possible, whereas the slow path offers reliable transmission and content caching that is essential for fast startup and recovery.

Our deployment and evaluation confirm the efficacy of these design choices (§6). LiveNet has acted as the foundation of Alibaba's low-latency streaming technology stack for 3 years. Compared with our prior hierarchical CDN, LiveNet condenses the average transmission path length (i.e., number of overlay hops) from 4 to 2 and reduces the delay between the ingress and egress points of the CDN by over 50%. It also significantly improves the user-perceivable experience: 95% of views have a startup delay of under 1s, and 98% have no stalls. After 3 years of operation, we believe that the design choices of LiveNet are not only feasible but also highly effective. The design choices of LiveNet and the lessons learned (§7) can be broadly applied to other large-scale streaming scenarios such as video conferencing and online education.

Our main contributions are as follows:

- We present LiveNet, a low-latency video transport network based on a flexible flat overlay topology, with centralized control for global optimization.
- We detail some of our key solutions for supporting low-latency live streaming, including our (*i*) centralized routing computation; (*ii*) fast path lookup and establishment; and (*iii*) a fast-slow path transmission architecture with fine-grained control on streams.
- We demonstrate the efficacy of LiveNet through Taobao Live, which is one of the biggest players in e-commerce live streaming. We also share our operational experiences of running LiveNet over the last 3 years.

We emphasize that LiveNet was born out of the practical needs identified during the operation of our previous generation video transmission network (Hier). Some of our design choices share similar ideas to existing work (such as flat CDN architectures [29, 37] and centralized control planes [9, 35]). That said, we present the details of the design and implementation of a mix of both existing and novel approaches (e.g., fast-slow path transmission system with fine-grained stream control) that can be employed to minimize the transmission delay of a large-scale commercial CDN system. More importantly, we verify this design in a production environment and share our experiences.

2 BACKGROUND AND MOTIVATION

2.1 Low-Latency Live Streaming

For many years, live streaming was primarily used for large event broadcasting (e.g., the World Cup). Here, the broadcasters are often the content providers themselves, and often it is acceptable to have an end-to-end latency in the order of several seconds [44].

More recently, platforms have enabled individuals to "go live", broadcasting their camera feeds to a global audience. This kind of live streaming is referred to as *personalized live streaming* and has more stringent latency requirements (as individual broadcasters may interact with their audience [36]). For example, as one of the world's largest online shopping apps, Alibaba Taobao offers a live streaming service (Taobao Live) to any online shops that would like to promote their products. As a live streaming service for e-commence, Taobao Live imposes strict latency and stability requirements as these could impact revenue. For instance, shops that sell similar products may co-live stream, in order to attract a larger viewership. In a co-stream, broadcasters can talk to others (video chat) and compete with each other, trying to win shoppers.

With the above in mind, we identify 3 key system requirements for low-latency live streaming: (*i*) It must offer low end-to-end latency for live video delivery (~1 second) and fast start-up (within 1 second); (*ii*) It must elegantly scale to tens of thousands of concurrent broadcasters and millions of viewers; and (*iii*) It must offer high

[1]This copied packet on the slow path will not be forwarded to the downstream nodes.

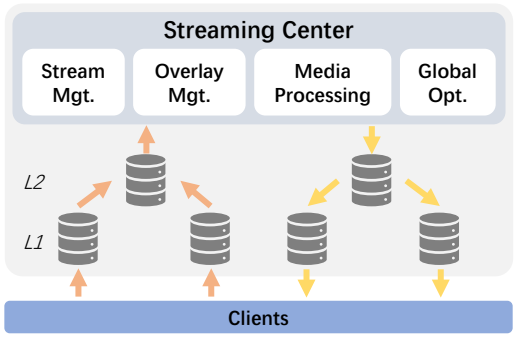


Figure 1: The system structure of Alibaba's first-generation hierarchical CDN (Hier).

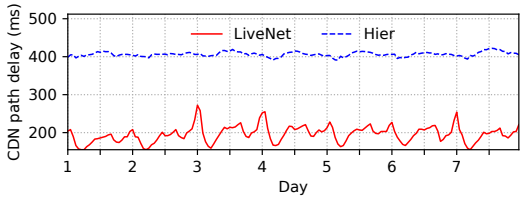


Figure 2: CDN path delay for Hier and LiveNet.

stability and deterministic performance, even during large-scale events.

2.2 Alibaba Hierarchical CDN (Hier)

Alibaba's first-generation video transport network follows the conventional wisdom of using a hierarchical CDN [32, 44], as shown in Figure 1.[2] We refer to this as Hier. It is composed of a powerful streaming center and geo-distributed CDN nodes, where each node consists of a cluster of machines in Alibaba Cloud. The streaming center undertakes almost all media data processing (e.g., video transcoding) and system management (e.g., CDN node management). While logically being centralized, it is geo-replicated in several data centers. CDN nodes, which are organized in a two-layer structure, are responsible for live video transport, but barely process video content. The nodes at a higher level are equipped with more bandwidth and storage resources. Note that, thanks to the global coverage of Alibaba Cloud, the L1 nodes (also called edges) are located close to end-users.

A typical live video stream in the hierarchical CDN is as follows: the broadcaster uploads their live content to the assigned L1 node, which then forwards the content to the streaming center via a selected L2 node. The streaming center transcodes the content if needed. The content is forwarded down to L2 and L1 nodes that are connected to viewers of this stream. L2 and L1 nodes may cache the content in the form of GoPs (Group of Pictures) to serve requests thereafter. The transport within the CDN overlay is based on RTMP over TCP. It is worth noting that we implemented a centralized control to coordinately map L1 nodes to L2 nodes for individual streams. The control is similar to VDN [35], where the

[2]Note that, it is still in production use at Alibaba.

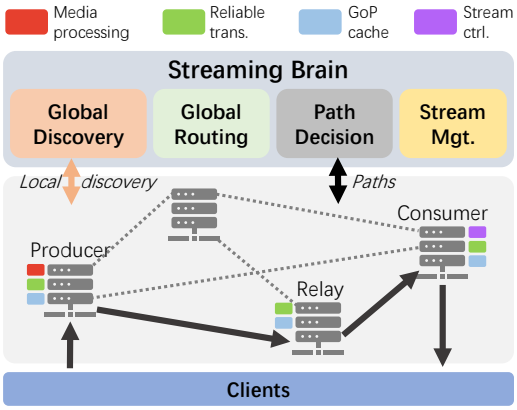


Figure 3: The system structure of LiveNet.

control component has a global view of the CDN overlay state and computes the map to optimize the predefined utility. By doing so, we avoid path congestion due to static mapping of L1 and L2 nodes and implement priority-based scheduling.

2.3 LiveNet: from Hierarchical to Flat CDN

We have operated Hier since 2016 and observed 3 important trends that motivate us to develop LiveNet. First, we have seen ever-increasing demands for improved user experience. For instance, we see many "flash sale" campaigns that are launched by online shop owners in Taobao Live. In such a campaign, limited products will be sold at extremely low prices during the live broadcast, following a first-come-first-serve policy. This requires sub-second end-to-end latency, which translates to an expected CDN delay of 300-400 ms.[3] However, as shown in Figure 2, the CDN path delay in Hier is ~400 ms. As such, we cannot make the SLA guarantee of such a low CDN delay to customers with Hier.

Second, we have seen continuous growth in the numbers of broadcasters and viewers in live streaming applications. Nevertheless, the rigid tree overlay topology means that all video streams have to go to the centralized media processing center, dramatically increasing the load on non-leaf nodes, and delaying the data delivery.

Finally, as a cloud provider, we also would like to offer a real-time video transport network to third parties. Supporting a diversity of use cases is therefore key to our business model, while the rigidity of Hier's overlay topology means that it is more difficult to handle things like traffic prioritization.

Inspired by the flexibility and efficiency offered by flat CDN overlays [29, 37] and centralized control [9, 33, 35, 37], we envisioned a low-latency video transport network, LiveNet, that is built upon a *flat* overlay network (see Figure 3). In LiveNet, each node's functionality as well as the overlay path between every pair of nodes can be configured through a centralized control center, called the *Streaming Brain*. To put this in context, Figure 2 plots the

[3]The customers that run live streaming services expect a 900-1000 ms end-to-end delay. This can be considered the minimum requirement for interactivity. This includes a ~300 ms playback buffer (to minimize stalls), ~150 ms for encoding and first-mile delay from broadcasters to the CDN, and ~150 ms for decoding and last-mile delay from the CDN to viewers.

CDN path delay of LiveNet. The typical delay lies between 150 ms and 250 ms. Thus, LiveNet allows us to provide video transport services that require an expected CDN delay of 300-400 ms. The rest of this paper details the design of LiveNet and the production performance achieved.

3 LIVENET OVERVIEW

We first provide a high-level overview of LiveNet. Later, we detail the centralized control plane (§4) before describing our stream forwarding plane (§5).

Design Goals. LiveNet aims at providing ***persistent low-delay*** (sub-second) live video delivery for ***large-scale*** concurrent broadcasters and viewers. It also requires ***flexibility*** in supporting various live streaming applications on one CDN overlay network.

End-to-End Workflow. End users (uploaders or viewers) are mapped to CDN servers via DNS redirection. The CDN node that a broadcaster connects to is called a *producer* node, and the CDN node that a viewer connects to is called a *consumer* node. A broadcaster uploads live content to the producer node, often using WebRTC, where the producer node processes the media content (e.g., transcoding) if needed. A consumer node will receive requests from viewers. If it is already serving that stream and has recent video frames cached, it will immediately respond with the content. Otherwise, it initiates a path lookup request to the central control plane with the stream ID as the input. The overlay path returned dictates how the live video content can be transmitted from the producer to the consumer nodes, potentially via intermediate nodes (which we refer to as *relays*). These relays cache video content and can be used to construct arbitrary path topologies based on diverse constraints. Our system is separated into a centralized control and decentralized data plane, which we briefly summarize below.

Control Plane. The centralized control plane (detailed in §4) collects the overlay network states from CDN nodes to form a global view of the overlay. Based on the global view, the controller computes the best paths (in terms of delay) for every pair of overlay nodes. The computation is naturally constrained by link capacities and node loads. Nevertheless, both constraints vary on a small time scale (e.g., seconds). This is because (*i*) live streams (i.e., channels) come and go often; (*ii*) views often last a short period; and (*iii*) overlay nodes may be also involved in video processing, leading to variations in loads. Our experiences have therefore led us to avoid a hybrid approach with centralized control and decentralized decisions [23, 35] because communication between the central and local controllers frequently becomes unaffordable. Instead, we focus the path computation and decision-making in a centralized controller, referred to as the *Streaming Brain*.

Data Plane. The data plane (detailed in §5) covers the transmission of data via the overlay paths that connect the CDN nodes. Our experience shows that the forwarding engine in the overlay nodes can impact performance significantly. For example, we have found that running a whole application stack (e.g., WebRTC) on each overlay node introduces unacceptable processing latency for our use cases. To address this challenge, LiveNet relies on a novel fast-slow path forwarding mechanism. This involves two parallel packet pipelines running on each single overlay node. First, upon receipt of a video packet, the *fast path* immediately forwards it downstream, circumventing the application stack. This means that such packets avoid any processing tasks such as congestion control, caching, or error detection. This is motivated by the fact that our network backbone is nearly lossless ($< 0.175\%$ even in peak hours).

Nevertheless, loss does occur during heavy loads, and a missing *I* frame severely impacts user experience. Thus, we supplement this with a second packet pipeline that we call the *slow path*. This pipeline operates a full-stack, including hop-by-hop transmission control to enable rapid packet recovery [43]. Therefore, when the fast path fails, consumer nodes can rapidly recover the loss. Specifically, the received packet is also copied to the slow path for congestion control, loss recovery, and GoP caching; nevertheless, the copied packet will not be forwarded to the downstream nodes. We delay providing comprehensive details until §5 but instead provide a brief example to show the benefits. Imagine a path: $A \rightarrow B \rightarrow C$. If no packet loss occurs, fast path forwarding will suffice. However, if B detects a loss (via the slow path), it will NACK A for retransmission. Thus, when C also detects the loss, the slow path may have already recovered the packet. In parallel, the fast path will continue to forward packets ensuring continuous delivery.

4 STREAMING BRAIN DESIGN

4.1 Overview

Taking inspiration from prior work in SDN, we rely on a logically centralized controller, referred to as the *Streaming Brain*. This is composed of four components, as shown in Figure 4. The *Global Discovery* module collects the overlay network state from CDN nodes, providing a global view for the path computation. The *Global Routing* module calculates the best paths for each pair of overlay nodes based on the global view periodically (every 10 minutes). These are then pushed to the *Path Decision* module, which serves path requests from consumer CDN nodes. This information is stored in the Path Information Base (PIB). Note that, the PIB is also updated by the *Global Discovery* module if any links or nodes become overloaded. Finally, when a CDN producer node receives a new stream uploading request, it will push the stream information (e.g., stream ID) to the *Stream Management* module which maintains a record of which streams are active, stored in the Stream Information Base (SIB).

4.2 Global Discovery Module

The Global Discovery module collects network states from overlay nodes. Currently, we consider link latency (RTT), packet loss rate, link utilization, and node load.[4] Individual nodes report these metrics on a 1-minute time scale. Specifically, for a link l that connects two nodes i and j, if node i transmits data within the last 10 minutes over link l, it reports the network statistics directly from the transport layer. Otherwise, node i randomly selects a machine in its cluster to actively measure the statistics of the link l using the UDP Ping utility to ping the node j.[5]

[4] A node's load is a combined metric reflecting the number of stream transmissions going through it, the CPU and memory utilization.
[5] On each measure, we send only a few MSS packets.

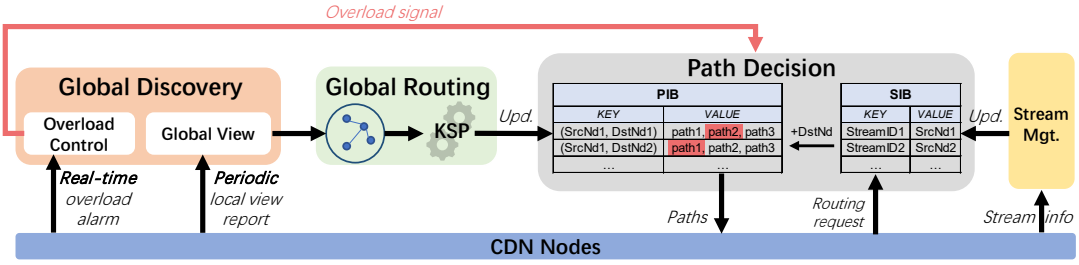


Figure 4: The Streaming Brain design overview.

Note that, our experiences have shown that link utilization and node loads may change on a relatively small time scale (e.g., seconds). If a link or node is about to be overloaded, the computed paths that involve them should potentially be invalidated. To reflect this, overlay nodes issue reports to the Global Discovery module if the node utilization or the utilization of their associated links reaches a pre-defined target (default 80%). The Global Discovery module then asks the Path Decision module to invalidate the corresponding paths that include the overloaded links or nodes.

4.3 Global Routing Module

The Global Routing module computes the routes for each pair of nodes every 10 minutes, with the up-to-date view of the overlay that is provided by the Global Discovery module.

Problem Formulation. Let $r \in R$ denote a viewing request, $path(r) \in P$ denote a specific path from the producer to the consumer node serving this viewing, and $Q(r, p)$ denote the expected delay for r when using p (a smaller value is better). We assume that the path selections for requests are independent, i.e., the performance of transmission is not impacted by the decisions made for other requests. Our goal is to minimize the total delay for all requests $r \in R$:

$$arg \min_{path \in P^R} \sum_{r \in R} Q(r, path(r)) \tag{1}$$

The above optimization is subjected to 3 constraints: (*i*) the traffic over a link is bounded by its bandwidth; (*ii*) the utilization of any node is bounded by a pre-defined target (e.g., 80%); and (*iii*) the path length should not exceed a pre-defined target (3 in our design)[6].

Two-Step Solution. Considering the performance demands of path computation and the ease of maintenance, LiveNet does not adopt an optimization-based approach (e.g., modeling overlay routing as a multi-commodity flow problem [9, 37]). Instead, LiveNet adopts a heuristic solution to determine the transmission path. Specifically, we abstract the link weights by incorporating link delay, packet loss rate, link utilization as well as the utilization of the two end nodes. For a link $A \rightarrow B$, its weight W_{AB} is abstracted as:

$$W_{AB} = (\rho \times 2 * RTT_{AB} + (1 - \rho) \times RTT_{AB}) \times f(u_{AB}) \tag{2}$$

$$f(u_{AB}) = 1/(1 + e^{\alpha * (\beta - u_{AB})}) + 1 \tag{3}$$

[6]We limit the path length to achieve low transmission delay and to reduce the number of locations where packet loss or bandwidth variations may occur.

where ρ is the packet loss rate of the link, and u_{AB} is the maximum value from the link utilization, A's node utilization, and B's utilization. The first term of Eq. 2 captures the expected link RTT, where we assume a lost packet can be recovered in the second attempt. Specifically, it is composed of the expected two-way delay when the packet is lost and recovered in the next try ($\rho \times 2 * RTT_{AB}$), as well as the delay when packet experience no loss ($(1 - \rho) \times RTT_{AB}$). $f(\cdot)$ is a Sigmoid-like function (ranging from 1 to 2) to adjust the weight, where α and β are hyper-parameters controlling the shape of $f(\cdot)$. We use $\alpha = 0.5$ and $\beta = 80\%$ in our current implementation.

With the above abstracted graph, we solve the global routing optimization problem through two steps. In the first step, we find the k ($k = 3$ in our current implementation) shortest paths between every pair of nodes using the K Shortest Paths (KSP) algorithm [19]. In the second step, we filter out any links that violate our constraints. Specifically, the paths that are longer than 3 hops or contain overloaded links or nodes are removed. The remaining paths are used by the Path Decision module.

Last-Resort Paths. In the case that all the computed paths between a pair of nodes violate the constraints, LiveNet provides last-resort paths. Specifically, we maintain a small quantity of last-resort nodes that are reserved only for last-resort paths. Any node can reach another node through one of these last-resort nodes in two hops. A last-resort path constitutes the producer node, a last-resort node as the relay, and the consumer node. To offer low-delay transmission, the last-resort nodes are located in networks that have many peering points with other networks (e.g., Internet eXchange Points, IXPs). In practice, around 2% of the viewing sessions are served using last-resort paths.

Supporting Other Applications. The above routing scheme is tailored for large-scale live streaming applications, especially Taobao Live. Nevertheless, our centralized computation means that the routing scheme or the associated constraints can be arbitrarily updated without impacting the CDN nodes. For instance, to support video telephony that requires lower delay but with a much smaller number of participants, the routing scheme can preferentially choose the CDN nodes that have good network connectivity and low loads as MCUs (Multipoint Conferencing Units). Detailed routing schemes and constraints for other applications are out of the scope of this paper.

Algorithm 1: Viewing request processing on consumer nodes

```
input : Stream ID: sid, Consumer: DstNd, and Client ID:
        ClientID contained in the viewing request
1  if sid ∈ StreamFIB then
2      StreamFIB[sid].append(ClientID);
3      return GoPCache[sid];
4  else
5      paths ← GetPath(sid, DstNd);
6      BestPath ← paths[0];
7      BackupPaths ← paths[1 :];
8      stream ← EstablishPath(sid, BestPath);
9      GoPCache[sid] = stream;
10 end
11 Function GetPath(sid, DstNd):
       // Executed at the Path Decision module.
12     SrcNd ← SIB[sid];
13     paths ← PIB[SrcNd][DstNd];
14     for p ∈ paths do
           // Delete paths with overloaded
              nodes/links.
15         if IsInvalid(p) == True then
16             paths.del(p);
17         end
18     end
19     return paths;
20 End Function
```

4.4 Path Decision Module

The Path Decision Module is responsible for selecting an overlay path for a given stream. Algorithm 1 elaborates how a consumer node responds to a viewing request with the Path Decision module.

Information Bases. The Path Decision module maintains a Stream Information Base (SIB) and a Path Information Base (PIB). The SIB stores for each live stream (indexed by a unique stream ID) its producer node in a hash table; it is updated when new streams are initialized or existing ones finish. The PIB, which is also a hash table, stores for each pair of nodes the candidate paths that can be leveraged for video content transmission from the producer to the consumer. The PIB is updated periodically by the output of the Global Routing module on a 10-minute time interval. Moreover, the Global Discovery module will immediately invalidate the paths in the PIB that contain overloaded links or nodes once it receives overload alarms from overlay nodes.

Path Lookup. Consumer nodes communicate with the Path Decision module for each overlay path lookup, using the stream ID as the input. This is first hashed to locate the producer node ID in the SIB. The producer ID is then combined with the consumer node ID as a key to locate the list of candidate paths in the PIB. In our current implementation, the returned list contains 3 paths ordered by their preference. As both information bases are built on hash tables, the path lookup takes only a few milliseconds.

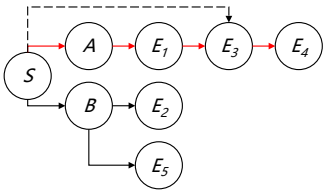


Figure 5: An illustration of the long-chain problem resulting from a cache hit.

In spite of the fast hash lookup, we would like to avoid excessive requests to the Path Decision module. Otherwise, it may become a performance bottleneck. Thus, LiveNet takes several steps to minimize this load. First, if a consumer node is already serving a stream to other viewers, it does not look up a new path upon receiving another view request. Second, when multiple requests for a popular stream arrive at the same time at a consumer node, the node only initiates one path lookup to the Path Decision module. Finally, for popular broadcasters,[7] up-to-date overlay paths are proactively pushed to all overlay nodes in advance of any viewers arriving.

Overlay Path Establishment. As indicated in Algorithm 1, the first element of the returned path list is considered the best candidate (line 5). This path will be preferentially established, while the other two are regarded as backup paths. Note that, LiveNet's architecture allows for any arbitrary policies to be employed for selecting the preferred path.

Upon retrieving a path, the consumer sends a request to the stipulated first hop on the reverse route towards the producer. If an intermediate relay node has already subscribed to this stream (i.e., called a *cache hit*), it will stop backtracking and forward the content to the consumer thereafter by adding it to its local Stream Forwarding Information Base (FIB). Note that because a prior path may have already been established, the resulting path from the producer to the consumer might be different from the one returned by the Path Decision module.

An undesirable consequence of the above is the *long-chain* problem. This is illustrated in Figure 5. Let us assume the path returned to the consumer node E_4 for this stream is $p = S \rightarrow E_3 \rightarrow E_4$ (2 hops). Now, imagine that E_3 has previously subscribed to this stream via $\hat{p} = S \rightarrow A \rightarrow E_1 \rightarrow E_3$. This triggers a cache hit in E_3, and the yielded path for E_4 becomes $p' = S \rightarrow A \rightarrow E_1 \rightarrow E_3 \rightarrow E_4$ (4 hops). To address this, consumer nodes monitor the quality reported by clients, and switch to an alternative path (i.e., the current best path provided by the Path Decision module) in case of poor quality. This is done if the number of stalls (or delay) exceeds a certain threshold that is set by the app.

5 LIVENET DATA TRANSMISSION

The previous section has shown how the Streaming Brain computes overlay paths for each stream. We next describe the transmission

[7] Broadcasters' popularity is measured by their historical viewing statistics. Large live streaming campaigns (which will often attract a large audience) may also notify us in advance.

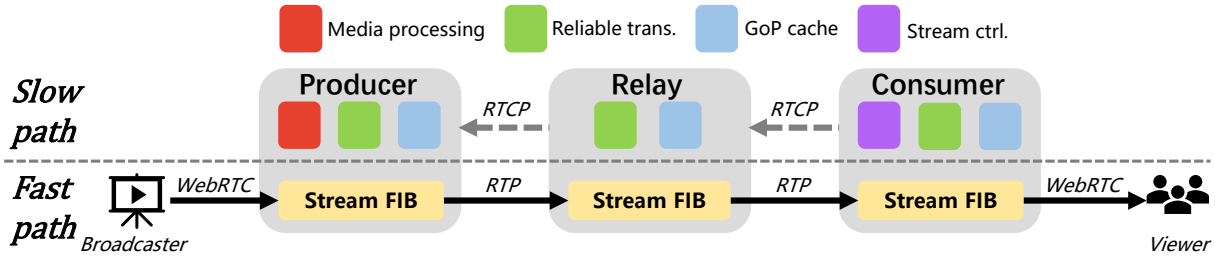


Figure 6: Transmission architecture overview.

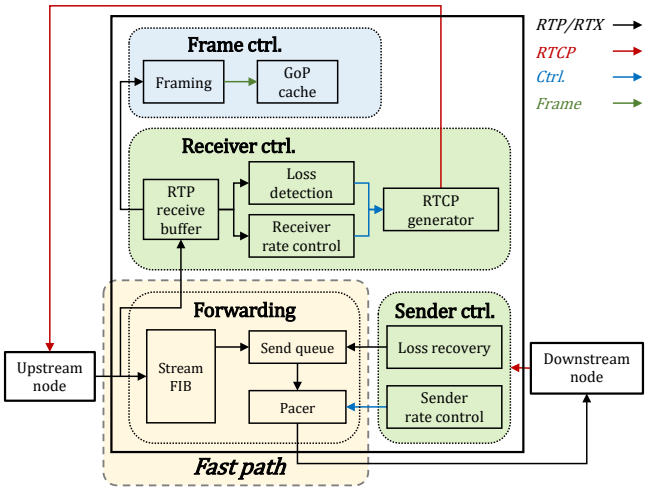


Figure 7: The software stack on overlay nodes for fast-slow path design.

process by which broadcast data is passed through the established overlay path from the producer to individual consumer nodes.

5.1 Transmission Architecture

A broadcaster is first mapped to a producer node via DNS, and then communicates with the producer over WebRTC. Upon receiving a request, a consumer node contacts the Streaming Brain to retrieve a valid path via its Path Decision module. As shown in Figure 7, each overlay node maintains a Stream FIB that records the subscriber nodes (i.e., downstream nodes to forward to) for each stream. The Stream FIB on a node is updated by the subscription and unsubscription requests from downstream nodes. For instance, in Figure 5, E_4 sends a subscription request to E_3, in order to subscribe to stream s_x. Upon receiving the subscription, E_3 adds a new forwarding rule to its local Stream FIB, $< s_x, \{E_4\} >$. If there is another node E_5 also wants to subscribe s_x from E_3, E_3 will update the rule as $< s_x, \{E_4, E_5\} >$.

With the above in mind, as illustrated in Figure 6, we adopt a novel *fast-slow path* design, consisting of two logical parallel packet processing pipelines. This design essentially separates the packet processing on a node into two paths: (*i*) packet forwarding on the fast path; and (*ii*) congestion control and loss recovery on the slow path. Thus, the fast and slow paths implement different functions provided by different protocol stack layers. On receiving an RTP packet, a node looks up the Stream FIB to get the next hop nodes (i.e., subscribers of this stream). It then immediately forwards the packet to all subscriber nodes in an optimistic "best-effort" way — it neither guarantees successful delivery of the packet, nor performs additional processing on the packet. The sending rate is controlled by the pacer module. This pipeline is referred to as the *fast path*.

A copy of the packet is replicated to the receiver buffer on the second pipeline (i.e., the *slow path*) in the meantime. Whereas the fast path is optimistic, assuming no packet loss, the slow path introduces caching, loss recovery, and congestion control via immediate backup and per-hop retransmission, in case the fast path experiences a loss. Specifically, the slow path adopts GCC [13] for congestion control: The sender rate control decides the pacing rate based on both the delay-based receiver-side control and the loss-based sender-side control. This pacing rate will then be passed to the pacer in the fast path, which will maintain inter-packet spacing at the time each packet is scheduled for transmission. The fast path is the executor of the congestion control strategy decided by the slow path.

For loss recovery, each node examines holes in the sequence numbers of the received RTP packets every 50 ms and sends the sequence numbers of the lost packets to the upstream node in RTCP NACK messages. The lost packets will then be retransmitted by the loss recovery module in the upstream node.[8] Note that both the fast and slow paths follow the same physical route.

Finally, on each node, the ordered packets in the slow path will be submitted to the Framing Control module, where packets are decoded into GoPs. The most recent GoPs are cached to facilitate fast startup of subsequent views on this stream.

5.2 Fine-Grained Control of Streams

Supporting Multiple Bitrates. There are two typical ways to support multiple bitrates of live video at the broadcaster side: SVC (scalable video encoding) and simulcast [10]. While SVC is appealing in video conferencing or video telephony [43] where the number of receivers is limited, it has a non-trivial encoding overhead (~10%) [43] and thus would consume significant extra bandwidth when serving millions of concurrent viewers. We thus adopt the simulcast mode, where the broadcaster encodes several different bitrate versions of the video (e.g., 720P+480P) in parallel and uploads all versions to the producer. The consumer nodes then evaluate

[8]The retransmitted packets have a higher sending priority than the packets in the send queue in the fast path.

each viewer's available bandwidth and select the best bitrate on its behalf. Note that each bitrate version of a stream corresponds to a unique stream ID in our system.

Seamless Stream Switching. Consumer nodes help clients *stream switch* within individual sessions. This occurs during co-streaming, where the broadcaster invites others to co-broadcast. Co-streaming essentially ceases the solo-broadcast stream (i.e., one of the broadcasters itself), and starts a new stream for co-broadcasting. Instead of asking viewer clients to resubscribe to the new stream, the consumer node will instead resubscribe to the new stream on the client's behalf. Thus, it automatically switches to forwarding the content of the co-broadcasting stream to the client once it has received a complete GoP of the stream. By doing so, the client will not perceive any playback stalls during stream switching. It also avoids embedding excess control functionality in the client.

Proactive Frame Dropping. Consumer nodes also impose other kinds of stream control. A particular case is that a consumer node can proactively drop frames in cases where its sending queue (per client) is building up too fast (exceeding an app-specific threshold). In this case, it will first drop unreferenced B frames [42]; dropping such frames only causes short blurring. If the queue still gets longer, P frames will be dropped, and finally the whole GoP. This proactive frame dropping is used to combat bandwidth variations, primarily in mobile networks [48]. In synergy, the consumer node will request a lower bitrate stream version if the sending queue is consistently building up.

Priority-Aware Data Sending. In the fast path, the data sending rate of an overlay node is controlled by a pacer, where the link bandwidth is estimated using GCC [13]. Inspired by WebRTC [13], we use a pacing gain of 1.5 when sending I frames because they are much larger than other P/B frames, in order to quickly empty the sending queue to avoid queuing delays. We also prioritize the queue containing audio packets (over video frames), in order to avoid head-of-line blocking by large video frames.

6 EVALUATION

LiveNet has deployed 600+ CDN nodes in more than 70 countries and regions around the globe. At present, LiveNet has been running in Alibaba Cloud for about 3 years, underpinning a number of large-scale live streaming applications. In this section, we evaluate LiveNet's performance in a production environment, as well as present a case study of the Double 12 shopping festival.

6.1 Methodology

We compare LiveNet with Hier using logs taken from *Taobao Live* video streams. We are in the middle of migrating from Hier to LiveNet, so currently they run in parallel. LiveNet and Hier share the same pool of CDN nodes in Alibaba Cloud, which does not distinguish whether a session is from LiveNet or Hier when allocating resources. They have similar footprints in terms of node locations, allocated resources, and imposed workloads. Note, Hier is representative of other state-of-the-art hierarchical solutions: (*i*) its architecture is similar to other commercial systems for live streaming (e.g., Facebook Live [32]); and (*ii*) it implements the VDN [35] concept, a customized controller for global optimization.

Table 1: Performance comparison of LiveNet and Hier. We report medians for the first 3 metrics.

	LiveNet	Hier	impr. %
CDN path delay (ms)	188	393	52.2%
CDN path length	2	4	50.0%
Streaming delay (ms)	948	1,151	17.6%
0-stall ratio (%)	98	95	3.1%
Fast startup ratio (%)	95	92	3.2%

Evaluation Data. We have collected performance data from Taobao Live, which ranks as the 2nd most popular app in China [3]. The data used in this paper spans 20 days in Dec. 2021, covering the Double 12 shopping festival [4], which is one of the two major online-shopping campaigns each year in Taobao Live. . We have three data sources.

The first is logged at CDN consumer nodes, where each log corresponds to a stream. It records the performance statistics of the used overlay path, including: (*i*) the path length in hops; (*ii*) the CDN path delay that includes both the propagation delay of each hop and the processing time on individual nodes that are involved in the path; (*iii*) the *first-packet delay* that records the time elapsed from receiving the viewing request to sending back the first data packet; and (*iv*) a *local hit indicator* showing whether the consumer has already got the path information for the requested stream.

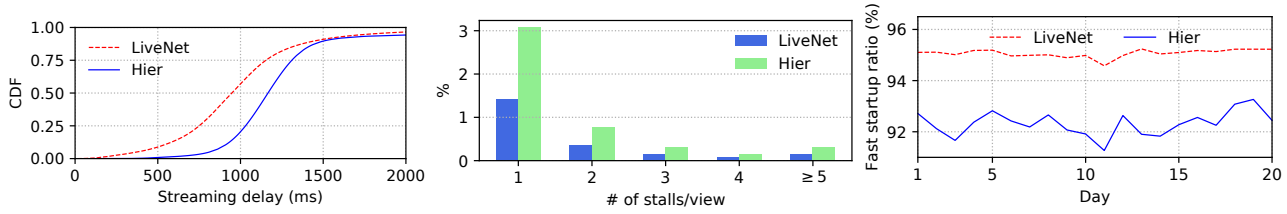
The second source is logged at individual clients. A log records the QoE statistics of each view, including: (*i*) the average *streaming delay*, which approximates the delay between the broadcaster's camera capturing a frame and the frame being displayed to the viewer. To measure this delay, we use a delay field in the RTP header extension. In the first packet of each *I* frame, the broadcaster adds the frame encoding time, packet queue time, and half of the next hop's RTT. Each intermediate node adds its processing time and half of the next hop's RTT to this field. The viewing client adds the client buffering time and frame decoding time, and outputs the final delay. Put simply, the streaming delay captures the sum of CDN path delay, first/last mile transmission delay, as well as the buffer time; (*ii*) user-perceived QoE metrics, including the number of *stalls*, which records how many times the playing buffer is vacant, and *fast startup indicator* shows whether the startup delay is within 1 sec.

The third source is logged at the Path Decision module in the Streaming Brain. Each log corresponds to a path request, and records the path request response time, i.e., the duration from receiving a path request to sending back the path list.

Ethics. Data is anonymized and approved by our institute's review board. It was collected as part of routine operations, and we do not link nor analyze individual client activities.

6.2 Overall Performance

We summarize the key performance statistics in Table 1 to compare LiveNet and Hier. The first three metrics are the medians over all viewing sessions, and the last two are the ratios over all sessions (across the 20-day observation period). Compared with Hier, LiveNet greatly improves the performance. We run a t-test, and obtain p-values < 0.001, which indicates the improvement is



(a) CDF of streaming delay. (b) % of views (y) that experience x stalls. (c) Fast startup ratio.

Figure 8: QoE metric comparison between LiveNet and Hier.

statistically significant. Specifically, the two CDN performance metrics are improved by 50%, and the average delay is reduced by 17% from 1.15s to 0.95s. Thus, LiveNet fulfills our requirement of sub-second streaming delay. We note that the difference between the improvements of the CDN path delay and the streaming delay comes from the fact that the streaming delay refers to the end-to-end delay: It is composed of CDN path delay, edge transmission delay (~300 ms), as well as a player buffer of fixed length (300 ms). Of the three parts, LiveNet can only optimize the CDN path delay. The improvement in fast startup ratio and 0-stall ratio (the ratio of the viewing sessions experiencing fast startup and 0-stalls respectively), while small, is of great importance. Note, our 3% improvement equates to millions of extra sessions experiencing no stalls because of LiveNet. We note the gap between the CDN path delay and streaming delay is due to the processing at clients and first/last mile transmission (i.e., the transmission between CDN nodes and clients).

6.3 QoE Performance

We next compare LiveNet with Hier using the QoE metrics recorded in the clients. Figure 8 shows the perceived streaming delay, number of stalls, and the fast startup ratio.

First, we see from Figure 8(a) that, in comparison with Hier, LiveNet improves the end-to-end streaming delay by over 200 ms for 60% of views, and by at least 100 ms for 80% of views. The reduction for the remaining 20% of views is small because of the limited coverage of the CDN edge nodes (i.e., producer and consumer nodes) near the client's location. In such cases, the first/last mile transmission dominates the streaming delay.

Second, Figure 8(b) shows the percentage of stalls that occur within a stream. Only 2% of the views have experienced at least one stall in LiveNet, while this is 5% for Hier. Among the views experiencing stalls, the vast majority (60%) only had one stall, whereas the probability of views with 5+ stalls in Hier is twice that of LiveNet. This constitutes a significant improvement to QoE.

Third, Figure 8(c) reports the fast startup ratio for each day during the observation period. Recall that, this represents the percentage of streams that start within 1 second. We see that the fast startup ratio of LiveNet consistently outperforms Hier (average 95% vs. 92%). Note that the small variations are due to overall load changes and also the scale-in and scale-out of underlying resources.[9] We note that the fast startup ratio is much higher than the fraction of views experiencing a streaming delay of less than 1s (60% in Figure 8(a)).

[9]We may use more machines during high-load.

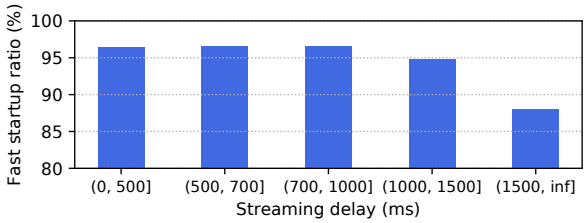


Figure 9: Fast startup ratio of LiveNet vs. different streaming delays.

This gap shows the effect of GoP caches in CDN nodes. To show this, we plot the fast startup ratio of views with different streaming delays in Figure 9. We see that the ratio is around 95% even when the streaming delay lies between 1s and 1.5s; this percentage is as high as 87% even when the streaming delay is over 1.5s.

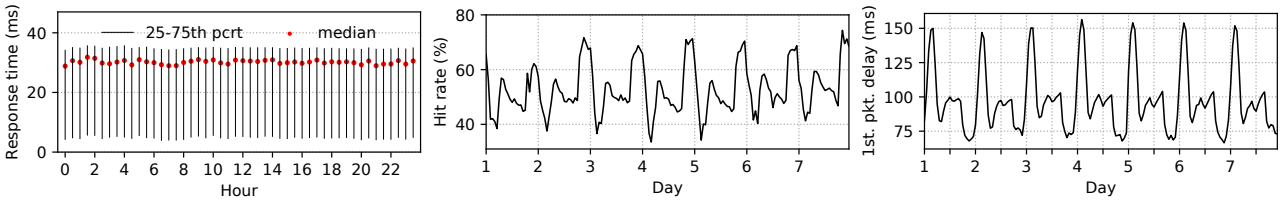
6.4 CDN Path Performance

We next explore the reasons for the prior performance improvements. Namely, we inspect our path computation and decision making, both of which exploit our move from a fixed hierarchical overlay to a more flexible flat model.

Streaming Brain Response Time. First, we measure the response time of the Streaming Brain. Figure 10(a) plots the response time for path requests to the Path Decision module. The median response time is only 30 ms, and the 25th percentile time is about 5 ms, confirming fast lookups.

Path Caching & Prefetching. Recall that we strive to minimize load on the Streaming Brain by caching path information on each node, as well as using prediction-based prefetching for paths that we predict to be popular or to be later requested by viewers (§4.4). When a path is already loaded into a node's Stream FIB, we refer to it as a *local hit*. Figure 10(b) plots the variation of the local hit ratio over a random week extracted from the dataset. We see a clear diurnal pattern, where the hit ratio reaches ≈70% between 8 pm and 11 pm when the load reaches its highest in a day. This suggests sufficient locality to make path caching worthwhile.

To test if this translates to improved performance, we compute the time between when a consumer node receives a viewing request and when it sends back the first packet. We plot the hourly average statistics in Figure 10(c) as a time series. Except for the period between 3 am to 6 am, when the local hit ratio is low (Figure 10(b)),



(a) Path request response time. (b) Path request hit ratio. (c) First-packet delay.

Figure 10: Evaluation of the Path Decision module and its impact on first-packet delay.

Table 2: CDN path length distribution for LiveNet.

	0	1	2	≥ 3
All	0.13%	7.00%	92.06%	0.81%
Inter-nation.	~0%	~0%	73.83%	26.16%
Intra-nation.	0.13%	7.16%	92.48%	0.23%

the first-packet delay is nearly always below 100 ms. In fact, the overall delay is as low as 70 ms between 8 pm to 11 pm, partially because of the high local hit ratio. The low first-packet delay is the key factor for the high fast startup ratio (95%). These results confirm the feasibility of relying on a centralized controller, alongside the benefits of preemptively placing decisions (paths) on the overlay nodes.

Path Length. One of the key design goals of LiveNet is to minimize delay by reducing CDN path lengths. Practical experience indicates that paths should ideally be under 3 (overlay) hops to attain the best performance. To evaluate our ability to ensure this, we compute the distribution of path lengths generated by LiveNet, shown in Table 2 (first row). Note, the path length for Hier is fixed at 4 (2 hops to the streaming center and 2 hops down to edge nodes).

90% of LiveNet's paths are 2 hops, and under 1% are 3+ hops.[10] This diversity is caused by the flat nature of LiveNet, allowing the Streaming Brain to compose arbitrary paths. To explore the path length diversity, we separate paths into inter-national and intra-national transmissions (last two rows in Table 2). Inter-national transmission is when the viewer and broadcaster reside in different countries (intra-national otherwise). We see that the proportion of long paths (i.e., ≥ 3) is significantly higher than that of intra-national transmission (26% vs. 0.23%). Nevertheless, the majority of inter-national and intra-national paths are still 2 hops (73% and 92%, respectively).

Path Delay. We next test if path reductions translate into lower CDN delays. Figure 11 plots the distribution of CDN path delays for different path lengths in LiveNet vs. Hier.

Unsurprisingly, the CDN path delay increases based on the number of overlay hops. 0-length paths offer the best performance, where a single overlay node acts as both the producer and consumer. In this case, the delay is solely the processing delay. While the overall trend shows a higher delay for longer paths, we see

[10]Note, while we limit the path length to 3 hops when computing routes, we may still encounter long chains (see § 6.4)

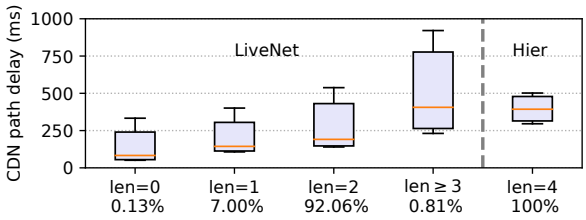


Figure 11: CDN path delay vs. path length in LiveNet and Hier. The box plot shows the 20th, 25th, 50th, 75th, and 80th percentile delay. The percentages on the x-axis indicate the proportion of paths.

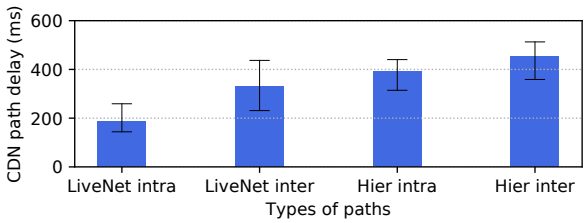


Figure 12: Path delay in inter/intra-national cases.

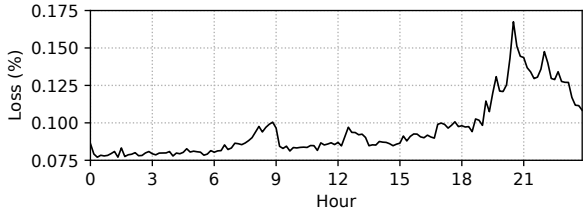


Figure 13: Temporal variation of average link packet loss rate (%) within the CDN.

cases where longer paths actually result in better delays. This occurs because the Streaming Brain's dynamic load balancing allows per-stream paths to bypass overloaded nodes.

Figure 12 further breaks down delays based on domestic vs. international paths. As expected, the CDN delay of intra-national transmission in LiveNet shows the best performance among all cases, the median path delay is under 200 ms (in contrast to 400 ms for Hier in intra-national cases). For the international transmission, the median is 330 ms for LiveNet and 450 ms for Hier.

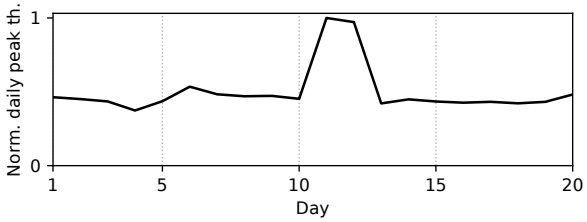


Figure 14: Peak system throughput from Dec. 1st to 20, normalized by the maximum daily peak throughput.

.

Table 3: LiveNet's performance during the Double 12 festival. We report medians for the first 3 metrics.

	Dec. 10	Dec. 11-12	Dec. 13
CDN path delay (ms)	188	192	180
CDN path length	2	2	2
Streaming delay (ms)	954	988	944
0-stall ratio (%)	97	97	97
Fast startup ratio (%)	94	94	95

Packet Loss Rate. Finally, Figure 13 reports the average packet loss rate of the overlay links over a day. We observe that although the loss rate rises in the peak hours (~9 pm), it is still under 0.175%, and the loss rate is less than 0.1% most of the time. This observation underpins our motivation for the fast-slow path transmission architecture.

6.5 Case Study: Double 12 Shopping Festival

We conclude by presenting a brief case study of a major live streaming event. The Double 12 shopping festival [4] is one of the largest shopping campaigns on Taobao held annually. It starts at 20:00 on Dec. 11th and ends at 23:59 on Dec. 12th, 2021. During this period, viewers scramble for the limited amount of discount goods, creating a significant spike in requests. Figure 14 depicts the normalized peak throughput per day, where we observe a significant spike during the festival (the peak throughput is doubled compared to a regular day).

We can now inspect how this load translates into CDN performance and QoE metrics. Table 3 presents the performance metrics between 10th to 13th Dec. Intuitively, one would expect these metrics to degrade during this significant extra load. However, there is no noticeable degradation observed, showing that LiveNet can effectively handle such aggressive spikes in demand. We highlight that this was achieved by an up-scaling in our provisioning during the festival (i.e., increasing the bandwidth resource cap for each node/link and leveraging more CDN nodes in Alibaba Cloud). We observe this in the number of unique overlay paths, which increased by 20% during Dec. 11th to 12th, in comparison with the days before and after.

7 DISCUSSION

7.1 Deployment Experiences

Streaming Brain Scalability. While logically centralized, the Streaming Brain is deployed on multiple geo-replicated data centers. Specifically, the Global Routing module along with the Global Discovery module is replicated in a few Alibaba data centers for reliability. We maintain consistency using a Paxos-like scheme [31]. Because the Path Decision module may impact stream startup delays, we replicate it in more locations to shorten the distances to consumer nodes. These locations include large AS networks (e.g., AS 4134) in which we deploy many CDN nodes as well as at IXPs (e.g., CNIX). Note that, replicas of the Path Decision module are updated by the Global Routing module.

Maintaining Multiple Paths. The Path Decision module in the Streaming Brain is configured to return 3 paths for each path request, ordered by their preference. However, during periods of heavy load, we have observed that path quality can degrade so rapidly that it is impossible for the Streaming Brain to update before streams have been affected. Thus, we enable consumer nodes to dynamically re-route paths based on local observations too. Specifically, consumer nodes establish *two* paths, with the highest priority one transmitting data and the second one only sustaining the connection as a backup. By doing so, consumer nodes can autonomously switch to the backup path when the primary one encounters a high delay or packet loss.

Mobility Support. Individual users may move during streams, and they may also switch between cellular and WiFi networks. If the optimal CDN consumer node of a viewer changes because of mobility, the client simply resubscribes to the stream through the new consumer node. The playback buffer at clients (e.g., 300 ms in Taobao Live) can help mitigate the possible stalls during mobility. In contrast, if the optimal producer CDN node changes due to broadcaster mobility, we wish to avoid having to update all prior stream paths in the CDN. Hence, the Streaming Brain instructs the old producer node to subscribe to the new one. By doing so, the existing overlay paths do not need to change.

7.2 Lessons learned

Flexibility Provided by LiveNet. LiveNet separates the control and data plane. The control plane is only responsible for the global optimization of routing, and is no longer involved in the end-to-end data transmission. The data plane, on the other hand, is in charge of media processing as well as the data transmission, following the configurations of the control plane. This separation offers greater flexibility as we can easily configure each overlay node to change the data path and the media processing logic. Further, because overlay nodes provide identical functionalities, we can easily circumvent the failed or overloaded nodes by migrating the tasks to others as instructed by the control plane.

Accelerating Playback Startup. While an extra path lookup may impact the startup delay of a view, 95% of viewing sessions in LiveNet still start within 1 second (see Table 1), and the average first-packet delay from consumer nodes receiving requests to sending back the first packet is as low as around 100 ms (see Figure 10(c)).

Our experience is that both the application and transport layers are equally important in accelerating playback. At the application layer, we leverage prediction-based overlay path pre-request and fine-controlled GoP caching. At the transport layer, we send I frames with a larger pacing gain (1.5). One of the interesting directions we are investigating is to leverage multi-path transport protocols (e.g., MPQUIC [48]) to improve the performance between overlay nodes and clients (i.e., the first/last mile transmission), which could further reduce the streaming delay.

A CDN for Multiple Services. Thanks to its flexibility, several low-latency live streaming services run on LiveNet, including Taobao Live, DingTalk video telephony, and several third-party apps (e.g., online classes).[11] These applications have different peak times in a day. For instance, DingTalk's peak time is during working hours, while Taobao Live's peak time is in the evening. By balancing these applications across the same underlying infrastructure, resource utilization can be improved and the cost can also be amortized.

Thin Clients. We have also learned to remove clients from the control loop as much as possible. For example, in LiveNet, the consumer nodes act as a delegate for each client's bitrate selection, as well as performing other services such as stream switching and frame drops. This has enabled us to more easily roll out new functionality, as well as simplify clients for deployment on heterogeneous devices.

7.3 Open Questions

There are several open questions we wish to highlight to further improve LiveNet. First, our two-step routing computation is heuristic and based on recent statistics of the CDN overlay. Data-driven routing [40] with load predictions and optimization-based solutions [9, 37] are worth investigating. Second, the flexibility provided by the flat CDN architecture creates challenges regarding capacity planning. We plan to experiment with deep learning-based overlay network planning [50]. This could use the overlay topology, the routing scheme, traffic demands, and cost models as an input. Finally, tailored congestion control algorithms for the transmission between overlay nodes may provide better performance than GCC. One of the interesting directions is to explore the benefits of online-learning-based congestion control [18].

8 RELATED WORK

Overlay Networks. Overlay networks have been studied extensively, including those optimized for low delay [17, 27]. In the context of content delivery, these were largely intended as an alternative to IP multicast [11, 14, 16, 30]. However, they often overlook the low latency requirement of live streaming and are, instead, oriented towards a peer-to-peer context [22, 47]. Conventional CDN overlays largely rely on multi-tier structures [44]. This structure has the benefit of enabling hierarchical caching, which can improve the QoE of VoD [8] and even live streaming [32, 41]. Our last generation CDN, Hier, uses a similar structure, but falls short of the stringent latency requirement imposed by low-latency live streaming like Taobao Live. While the idea of a flat CDN architecture with centralized path computation has been proposed before [29, 37], we augment it with our novel transmission architecture (i.e., the fast-slow path with fine-grained frame control).

Overlay Routing. Overlay routing has been investigated since the early 2000s, when the idea of centralized overlay routing using optimization-based methods was proposed [9]. There has been recent work revisiting overlay routing for video delivery [34]. VDN [35] proposes a hybrid centralized+distributed control for live video routing in CDN overlays. In contrast to us, this work overlooked fast-slow path forwarding on CDN nodes as covered in LiveNet. VIA [25] exploits the opportunity of relaying selective VoIP calls over overlay servers for better performance. While LiveNet also exploits relay paths, we mainly target large-scale live streaming. Prior approaches that map users to optimal CDN nodes (or edges), using either DNS [15] or anycast [20, 46], are complementary to our work.

Video Control Plane. We contribute to the wider literature on video control planes. Liu *et al.* explore the potential of global video control planes in improving video QoE [33] via a guided selection of suitable CDNs and video bitrates. C3 [23] implements a video control plane that assigns clients to suitable CDNs. CFA [26] is a data-driven approach that can be used by these video control planes for global optimization. In contrast, the centralized control plane in LiveNet interacts with overlay nodes (as opposed to clients) for overlay node configuration.

Low-Latency Video. Many have proposed video codecs to decrease end-to-end latency. Salsify [21] relies on a specialized codec to enable better interaction between the codec and the transport layer. Some other works [45, 49] advocate using deep learning to coordinate the codec and the transport layer for bitrate selection. LiveNAS [28] and Dejavu [24], on the other hand, enhance the live streaming quality by applying neural super-resolution at ingress nodes (i.e., producers in LiveNet). These optimizations focus on the first mile of live streaming and thus are orthogonal to our work.

9 CONCLUSION

This paper has presented LiveNet, a transport network for large-scale low latency live streaming. LiveNet includes a centralized Streaming Brain that computes routes based on a global view of the network, and a flat CDN overlay that implements fast content transmission and fine-grained control of frames. We have detailed the design of each component, and compared LiveNet with Hier, a state-of-the-art solution based on hierarchical CDN overlays. We have shown the superior performance of LiveNet and discussed the lessons we have learned while running LiveNet over the last 3 years. We hope our experiences can foster further research in this area to offer improved QoE for large-scale live streaming.

ACKNOWLEDGMENT

The authors would like to thank the shepherd Ramesh Sitaraman and the anonymous reviewers for their constructive comments. We thank all teams at Alibaba that help deploy LiveNet. Zhenyu Li's work was partially supported by National Key R&D Program of China (2019YFB1802800), Beijing Natural Science Foundation (JQ20024), and Natural Science Foundation of China (U20A20180, 62072437).

[11] We use different routing schemes and impose different constraints on the computed overlay paths for each one.

REFERENCES

[1] Youku lands domestic broadcast rights for 2018 world cup. http://www.ecns.cn/business/2018-05-30/detail-ifyuurnp0995123.shtml, 2018.

[2] 2020 global networking trends report. https://www.cisco.com/c/dam/m/en_us/solutions/enterprise-networks/networking-report/files/GLBL-ENG_NB-06_0_NA_RPT_PDF_MOFU-no-NetworkingTrendsReport-NB_rpten018612_5.pdf, 2020.

[3] Top mobile apps & platforms in china 2021. https://www.chinainternetwatch.com/30778/top-mobile-apps/, 2021.

[4] 12.12 sale : All you need to know about 1212 sale (double 12). https://cedcommerce.com/blog/all-you-need-to-know-about-double-12/, 2022.

[5] Broadcast live solution for global live streaming - alibaba cloud. https://www.alibabacloud.com/solutions/broadcast-live, 2022.

[6] Key takeaways from the taobao live 2021 livestreaming report. https://chinamktginsights.com/key-takeaways-from-the-taobao-live-2021-livestreaming-report/, 2022.

[7] Taobao live homepage. https://taolive.taobao.com/, 2022.

[8] V. K. Adhikari, Y. Guo, F. Hao, V. Hilt, Z.-L. Zhang, M. Varvello, and M. Steiner. Measurement study of netflix, hulu, and a tale of three cdns. *IEEE/ACM Trans. Netw.*, page 1984–1997, 2015.

[9] K. Andreev, B. M. Maggs, A. Meyerson, and R. K. Sitaraman. Designing overlay multicast networks for streaming. In *Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures*, pages 149–158, 2003.

[10] Z. Avramova, D. De Vleeschauwer, K. Spaey, S. Wittevrongel, H. Bruneel, and C. Blondia. Comparison of simulcast and scalable video coding in terms of the required capacity in an iptv network. In *Packet Video 2007*, pages 113–122, 2007.

[11] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *Proceedings of the ACM SIGCOMM 2002 Conference*, page 205–217, 2002.

[12] D. S. Berger, R. K. Sitaraman, and M. Harchol-Balter. AdaptSize: Orchestrating the hot object memory cache in a content delivery network. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 483–498, 2017.

[13] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo. Analysis and design of the google congestion control for web real-time communication (webrtc). In *Proceedings of the 7th International Conference on Multimedia Systems*, pages 1–12, 2016.

[14] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth multicast in cooperative environments. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, page 298–313, 2003.

[15] F. Chen, R. K. Sitaraman, and M. Torres. End-user mapping: Next generation request routing for content delivery. In *Proceedings of the ACM SIGCOMM 2015 Conference*, page 167–181, 2015.

[16] Y. Chu, S. Rao, S. Seshan, and H. Zhang. Enabling conferencing applications on the internet using an overlay muilticast architecture. In *Proceedings of the ACM SIGCOMM 2001 Conference*, page 55–67, 2001.

[17] J. Dai, Z. Chang, and S.-H. G. Chan. Delay optimization for multi-source multi-channel overlay live streaming. In *2015 IEEE international conference on communications (ICC)*, pages 6959–6964. IEEE, 2015.

[18] M. Dong, T. Meng, D. Zarchy, E. Arslan, Y. Gilad, B. Godfrey, and M. Schapira. PCC vivace: Online-Learning congestion control. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 343–356, Renton, WA, Apr. 2018. USENIX Association.

[19] D. Eppstein. Finding the k shortest paths. *SIAM Journal on computing*, 28(2):652–673, 1998.

[20] A. Flavel, P. Mani, D. Maltz, N. Holt, J. Liu, Y. Chen, and O. Surmachev. Fastroute: A scalable load-aware anycast routing architecture for modern cdns. In *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*, pages 381–394, 2015.

[21] S. Fouladi, J. Emmons, E. Orbay, C. Wu, R. S. Wahby, and K. Winstein. Salsify: Low-Latency network video through tighter integration between a video codec and a transport protocol. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 267–282, 2018.

[22] M. J. Freedman. Experiences with coralcdn: A five-year operational view. In *NSDI*, pages 95–110, 2010.

[23] A. Ganjam, F. Siddiqui, J. Zhan, X. Liu, I. Stoica, J. Jiang, V. Sekar, and H. Zhang. C3: Internet-Scale control plane for video quality optimization. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, page 131–144, 2015.

[24] P. Hu, R. Misra, and S. Katti. Dejavu: Enhancing videoconferencing with prior knowledge. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*, page 63–68, 2019.

[25] J. Jiang, R. Das, G. Ananthanarayanan, P. A. Chou, V. Padmanabhan, V. Sekar, E. Dominique, M. Goliszewski, D. Kukoleca, R. Vafin, and H. Zhang. Via: Improving internet telephony call quality using predictive relay selection. In *Proceedings of the 2016 ACM SIGCOMM Conference*, page 286–299, 2016.

[26] J. Jiang, V. Sekar, H. Milner, D. Shepherd, I. Stoica, and H. Zhang. CFA: A practical prediction system for video QoE optimization. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, page 137–150, 2016.

[27] S. Kaune, K. Pussep, C. Leng, A. Kovacevic, G. Tyson, and R. Steinmetz. Modelling the internet delay space based on geographical locations. In *2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, pages 301–310. IEEE, 2009.

[28] J. Kim, Y. Jung, H. Yeo, J. Ye, and D. Han. Neural-enhanced live streaming: Improving live video ingest via online learning. In *Proceedings of the ACM SIGCOMM 2020 Conference*, page 205–217, 2020.

[29] L. Kontothanassis, R. Sitaraman, J. Wein, D. Hong, R. Kleinberg, B. Mancuso, D. Shaw, and D. Stodolsky. A transport layer for live streaming in a content delivery network. *Proceedings of the IEEE*, 92(9):1408–1419, 2004.

[30] D. Kostić, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: High bandwidth data dissemination using an overlay mesh. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, page 282–297, 2003.

[31] L. Lamport. The part-time parliament. *ACM Transactions on Computer Systems 16, 2 (May 1998), 133-169. Also appeared as SRC Research Report 49. This paper was first submitted in 1990, setting a personal record for publication delay that has since been broken by [60].*, May 1998. ACM SIGOPS Hall of Fame Award in 2012.

[32] F. Larumbe and A. Mathur. Under the hood: Broadcasting live video to millions. https://engineering.fb.com/2015/12/03/ios/under-the-hood-broadcasting-live-video-to-millions/, 2015.

[33] X. Liu, F. Dobrian, H. Milner, J. Jiang, V. Sekar, I. Stoica, and H. Zhang. A case for a coordinated internet video control plane. In *Proceedings of the 2012 ACM SIGCOMM Conference*, page 359–370, 2012.

[34] B. M. Maggs and R. K. Sitaraman. Algorithmic nuggets in content delivery. *SIGCOMM Comput. Commun. Rev.*, 45(3):52–66, jul 2015.

[35] M. K. Mukerjee, D. Naylor, J. Jiang, D. Han, S. Seshan, and H. Zhang. Practical, real-time centralized control for cdn-based live video delivery. In *Proceedings of the 2015 ACM SIGCOMM Conference*, page 311–324, 2015.

[36] A. Raman, G. Tyson, and N. Sastry. Facebook (a) live? are live social broadcasts really broad casts? In *Proceedings of the 2018 world wide web conference*, pages 1491–1500, 2018.

[37] R. K. Sitaraman, M. Kasbekar, W. Lichtenstein, and M. Jain. Overlay networks: An akamai perspective. *Advanced Content Delivery, Streaming, and Cloud Services*, 51(4):305–328, 2014.

[38] Z. Song, D. S. Berger, K. Li, and W. Lloyd. Learning relaxed belady for content distribution network caching. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 529–544, 2020.

[39] A. Sundarrajan, M. Kasbekar, R. K. Sitaraman, and S. Shukla. Midgress-aware traffic provisioning for content delivery. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, pages 543–557, 2020.

[40] A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar. Learning to route. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, page 185–191, 2017.

[41] B. Wang, X. Zhang, G. Wang, H. Zheng, and B. Y. Zhao. Anatomy of a personalized livestreaming system. In *Proceedings of the 2016 Internet Measurement Conference*, page 485–498, 2016.

[42] Wikipedia contributors. Group of pictures — Wikipedia, the free encyclopedia, 2021. [Online; accessed 24-January-2022].

[43] Y. Xu, C. Yu, J. Li, and Y. Liu. Video telephony for end-consumers: Measurement study of google+, ichat, and skype. In *Proceedings of the 2012 Internet Measurement Conference*, page 371–384, 2012.

[44] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li. Design and deployment of a hybrid cdn-p2p system for live video streaming: Experiences with livesky. In *Proceedings of the 17th ACM International Conference on Multimedia*, page 25–34, 2009.

[45] H. Zhang, A. Zhou, J. Lu, R. Ma, Y. Hu, C. Li, X. Zhang, H. Ma, and X. Chen. *OnRL: Improving Mobile Video Telephony via Online Reinforcement Learning*. 2020.

[46] X. Zhang, T. Sen, Z. Zhang, T. April, B. Chandrasekaran, D. Choffnes, B. M. Maggs, H. Shen, R. K. Sitaraman, and X. Yang. Anyopt: Predicting and optimizing ip anycast performance. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, SIGCOMM '21, page 447–462, New York, NY, USA, 2021. Association for Computing Machinery.

[47] M. Zhao, P. Aditya, A. Chen, Y. Lin, A. Haeberlen, P. Druschel, B. Maggs, B. Wishon, and M. Ponec. Peer-assisted content distribution in akamai netsession. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, page 31–42, New York, NY, USA, 2013. Association for Computing Machinery.

[48] Z. Zheng, Y. Ma, Y. Liu, F. Yang, Z. Li, Y. Zhang, J. Zhang, W. Shi, W. Chen, D. Li, Q. An, H. Hong, H. H. Liu, and M. Zhang. Xlink: Qoe-driven multi-path quic transport in large-scale video services. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, SIGCOMM '21, page 418–432, New York, NY, USA, 2021. Association for Computing Machinery.

[49] A. Zhou, H. Zhang, G. Su, L. Wu, R. Ma, Z. Meng, X. Zhang, X. Xie, H. Ma, and X. Chen. Learning to coordinate video codec with transport protocol for mobile video telephony. In *The 25th Annual International Conference on Mobile Computing and Networking*, 2019.

[50] H. Zhu, V. Gupta, S. S. Ahuja, Y. Tian, Y. Zhang, and X. Jin. Network planning with deep reinforcement learning. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, page 258–271, 2021.