LATENCY EQUALIZATION AS A NEW
NETWORK SERVICE PRIMITIVE

Abstract:

Multiparty interactive network applications such as teleconferencing, network gaming, and online trading are gaining popularity. In addition to end-to-end latency bounds, these applications require that the delay difference among multiple clients of the service is minimized for a good interactive experience. We propose a Latency Equalization (LEQ) service, which equalizes the perceived latency for all clients participating in an interactive network application. To effectively implement the proposed LEQ service, network support is essential. The LEQ architecture uses a few routers in the network as hubs to redirect packets of interactive applications along paths with similar end-to-end delay. We first formulate the hub selection problem, prove its NP-hardness, and provide a greedy algorithm to solve it. Through extensive simulations, we show that our LEQ architecture significantly reduces delay difference under different optimization criteria that allow or do not allow compromising the per-user end-to-end delay. Our LEQ service is incrementally deployable in today’s networks, requiring just software modifications to edge routers.

Existing system:

Client-side latency compensation techniques:

It is based on hardware and software enhancements to speed up the processing of event updates and application rendering. These techniques cannot compensate for network-based delay differences among a group of clients. Buffering event update packets at the client side is hard to implement because this requires the coordination of all the clients regarding which packets to buffer and for how long. This leads to additional measurement and communication overhead and increased application delay.
Some gaming clients implement *dead reckoning*, a scheme that uses previously received event updates to estimate the new positions of the players. Dead reckoning has the drawback that the prediction error increases significantly with increasing network delays.

**Server-side latency compensation techniques:**

Due to the problems of client-side solutions, several delay compensation schemes are implemented at the server side. However, while introducing CPU and memory overhead on the server, they still do not completely meet the requirements of fairness and interactivity.

**Bucket synchronization mechanism:**

The received packets are buffered in a bucket, and the server calculations are delayed until the end of each bucket cycle. The performance of this method is highly sensitive to the bucket (time window) size used, and there is a tradeoff between interactivity versus the memory and computation overhead on the server.

**Time warp synchronization scheme:**

Snapshots of the game state are taken before the execution of each event. When there are late events, the game state is rolled back to one of the previous snapshots, and the game is re-executed with the new events. This scheme does not scale well for fast-paced, high-action games because taking snapshots on every event requires both fast computation and large amounts of fast memory, which is expensive.

A game-independent application was placed at the server to equalize delay differences by constantly measuring network delays and adjusting players’ total delays by adding artificial lag. Using server-based round-trip-time measurements to design latency compensation across players fails in the presence of asymmetric latencies.

**Proposed system:**

In this paper, we design and implement *network-based* Latency EQualization (LEQ), which is a service that Internet service providers (ISPs) can provide for various interactive
network applications. Network-based LEQ service provides equalized-latency paths between the clients and servers by redirecting interactive application traffic from different clients along paths that minimize their delay difference.

LEQ architecture provides a flexible routing framework that enables the network provider to implement different delay and delay difference optimization policies in order to meet the requirements of different types of interactive applications.

To achieve LEQ routing, we formulate the hub selection problem, which decides which routers in the network can be used as hubs and the assignment of hubs to different client edge routers to minimize delay difference. We prove that this hub selection problem is NP-hard and in approximable. Therefore, we propose a greedy algorithm that achieves equalized-latency paths. Through extensive simulation studies, we show that our LEQ routing significantly reduces delay difference in different network settings (e.g., access network delay and multiple administrative domains).

**Hardware requirements:**

- Processor : Any Processor above 500 MHz
- Ram : 1 GB.
- Hard Disk : 10 GB.
- Compact Disk : 650 Mb.
- Input device : Standard Keyboard and Mouse.

**Software requirements:**

- Technology : Net Beans 6.8
  - Jdk1.7