pVD - Personal Video Distribution

Fei Su, John Markus Bjørndalen, Phuong Hoai Ha, Otto J. Anshus
Department of Computer Science
University of Tromsø, Norway
fei.su@uit.no, jmb@cs.uit.no, phuong@cs.uit.no, otto@cs.uit.no

***Abstract*—A user has several personal computers, including mobile phones, tablets, and laptops, and needs to watch live camera feeds from and videos stored at any of these computers at one or more of the others. Industry solutions designed for many users, computers, and videos can be complicated and slow to apply. The user must typically rely on a third party service or at least log in. The Personal Video Distribution (pVD) system supports sending and viewing live and stored videos between any of a single user's computers, and allows for a smooth hand-over of play back between computers. The system avoids any third parties, and relies only on the user's personal computers. We present the architecture, design and implementation of the pVD prototype. The architecture is comprised of functionality for sending videos, subscribing to videos, and maintaining the video play-back state. The design has a local side sending and viewing videos, and a global side coordinating the switching and distribution of videos, and maintaining subscriptions and video state. The prototype is primarily done in Python. A set of experiments was conducted to document the performance of the prototype. The results show that pVD global side has low CPU usage, and supports a handful of simultaneous exchanges of videos on a wireless network.**

Keywords*—*Personal Video Distribution; Video Playback; Mobile Video Streaming; Video Hand-off.

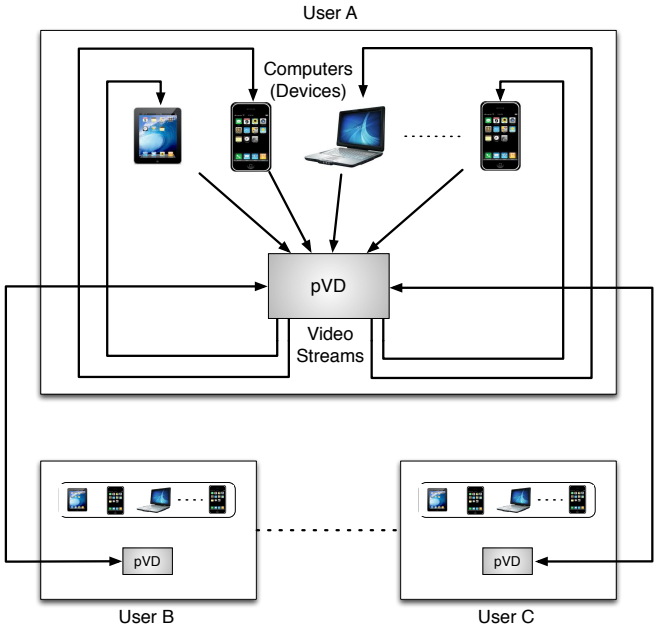

Fig. 1. The idea of the personal Video Distribution (pVD).

I. Introduction

Users today use multiple personal computers, including both mobile devices and larger displays. Many of these computers will have cameras that can be used to produce live video streams, and will have significant storage for videos. Live video from a camera connected to a computer can easily be watched on the same computer. This is also the case for videos stored on the computer. However, it is more cumbersome to do a smooth hand-over and watch video produced and stored at one of the user's computers at the others. It is also cumbersome to locate a video across computers because there is not a shared video name space. Consequently, videos at different computers can have the same name.

Existing industry approaches typically rely on a third party to let a user watch cameras and videos across computers. At the minimum, a log in to a subscription service is needed. As well as being dependent upon third party computers, an external network giving access to the Internet is also needed even when all video producing and consuming computers are local, say, at a user's home. This increases the probability for failures as well as cost and bandwidth usage.

Security and privacy are also issues users are concerned about when relying on third party services to store and service data [1], [2]. In [3] it is documented that people are more concerned about the privacy on mobile phones than laptops.

We report on the architecture, design and implementation of the Personal Video Distribution (pVD) prototype, allowing computers belonging to a single user to subscribe to cameras and videos from each other, see Figure 1. The system allows the live video from any camera to be viewed at any computer. When a stored video is played back or stored on one of the computers, it can be picked up by any of the computers. Play-back can be started from where in the video the user last stopped watching it. The pVD system does not rely on a third party service at all, using only a user's computers. When all computers are inside the same domain, say, at home, no Internet access is needed.

The usage model assumes that a user has physical access to all the computers. To watch, say, a smart phone's camera on a tablet, the user starts the pVD smart phone app and selects the camera as a video source. The app then starts streaming the video to the pVD system. On the tablet, the user starts the pVD app, inputs the smart phone's name and name of the live video, and a subscription is sent to the pVD system. The pVD app on the tablet now waits for videos to arrive according to the subscriptions. The pVD system matches incoming video streams with subscriptions, and streams the video to the tablet. While a single video can easily be streamed in and out of some smart phones and tablets, streaming of multiple videos are more easily supported using more powerful computers such

as laptops and PCs.

The prototype is presently functional for a single user with multiple computers. We intend to extend the prototype to support multiple users on a single pVD system, e.g. a family, in the future. We briefly describe how multiple users with separate pVD systems can share cameras and videos without relying on a third party service.

Our contribution is to document a flexible and simple way to switch videos between a single user's computers by using only these computers, and without relying on a third party (cloud) service at all. We document how to do this through the architecture, design and implementation of an actual working prototype, and its performance characteristics. Several experiments have been conducted to measure how fast the pVD system responds to subscription requests, the CPU utilization of the part of the pVD system used by all the computers, and how pVD scales when the number of videos and computers increase.

II. Related Literature

There are many existing live video streaming services. PPStream [4] and PPTV [5] are systems for video distribution over the Internet using a combination of client/server and peer-to-peer approaches for distribution of videos. They maintain the state of a user on the user's computer, including which videos the user watched and where the user stopped the playback of a video. The user can later start a video from where it was stopped. Contrary to pVD, these systems are large-scale sharing many videos between many users, and rely on every user having Internet access. In pVD, a computer also has the complete video so no peer-to-peer collaboration is needed. This reduces the complexity of the system.

YouTube is the world largest video sharing website from where people can upload, view and share videos. Live streaming is possible through services such as YouTube Live Streaming Events and Google+Hangouts. These systems are storing and sharing many videos between many users, while pVD shares just a few videos between a single user's computers. Users are dependent upon YouTube and Google as third parties outside of the users' control.

LiveCast [6] and Qik [7] enable live video streaming from users' mobile and other devices to any users or friends connected to the web. LiveCast is large scale with many users. It is feature rich, and meant to be used across Internet. Users are not dependent upon a third-party except their own organization or company. Qik is also on a large scale, enabling sharing between many users. The user must rely on Qik as a third party and store videos with Qik.

The Digital Living Network Alliance (DLNA) [9] uses Universal Plug and Play (UPnP) [8] for media management and media sharing between devices. Windows, Mac, Linux and Android also use the UPnP protocol to enable media sharing between devices. DLNA systems typically apply a media server and media players. In contrast, in pVD every computer is both a media server and a media player.

Apple AirPlay allows limited wireless streaming between Apple computers. The computers must be on the same subnet. While a video stream stored locally on one computer can be sent to another local computer, all computers must log into and interact with a third party, an Apple iTunes account.

A mobile live video learning system used for large-scale learning is described in [10]. Students can either attend a course in person or watch live and stored video streams sent from a server to mobile devices. The system does not maintain a user's video state to let play back resume at another computer.

Tele-TASK [11] is a tool for recording lectures and what happened at the presenter's computer, including presentation slides and software demonstrations. Users with mobile devices can watch the lectures everywhere.

CloudPP [12] is a Cloud-Based P2P Live Video Streaming Platform. It uses third-party cloud servers to construct a video delivery platform.

LiveShift [13] streams both live and stored videos. Live videos are streamed through a P2P network and peers store received videos for future playback. A user can view a stored video without local recording, and jump over boring parts to catch up the live video.

Eunsam Kim et al. [14], proposed an on-demand TV service architecture for a networked personal video recorder. This design reduces interactive operation response time and saves network bandwidth. The architecture includes origin servers, cache servers and Networked PVRs. The system serves both live videos and stored videos for playback.

Mobicast [15] is a mobile live video streaming system. Multiple users stream the same event from their devices. The streams can be stitched together, or the stream that has a best viewing angle is selected to provide a better collective viewing effect to viewers. If two users stream the same view, one of the streams can be stopped and later resumed to conserve battery power on the mobile device.

The systems mentioned above share to a large extent some characteristics. They are typically making a client dependent upon a third party outside of the client's control. They are intended for very many clients. We expect them to be rather complicated because they need P2P and other approaches to maintain good performance when the numbers of clients grow. While they all allow a user to playback videos, these videos are in most cases not meant to be on the user's computers. The ability to switch a live camera feed between a user's computers is only available in a few of the systems.

A significant characteristic of pVD is that a single user's computers subscribe to video streams from each other. Multiple subscriptions can be set up. However, a producer of a stream and the streams subscribers does not have to be running at the same time. When a stream starts streaming it can be picked up, and when no subscribers are running, the stream will go to the pVD system where it is buffered until a subscriber becomes present. pVD can also do hand-over of video streams between computers, letting a video start playing again from where it was stopped at another computer. The state of all videos are stored at and handled by the pVD global side server, and not by the pVD local side computers.

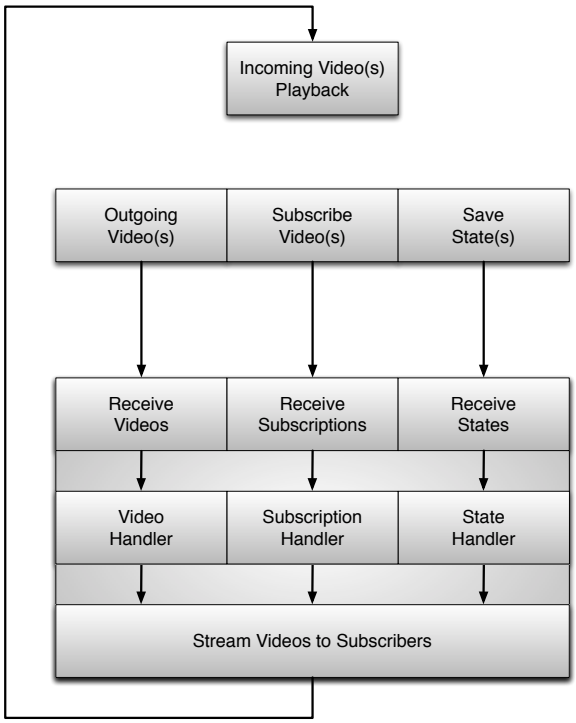

Fig. 2. The architecture of pVD.

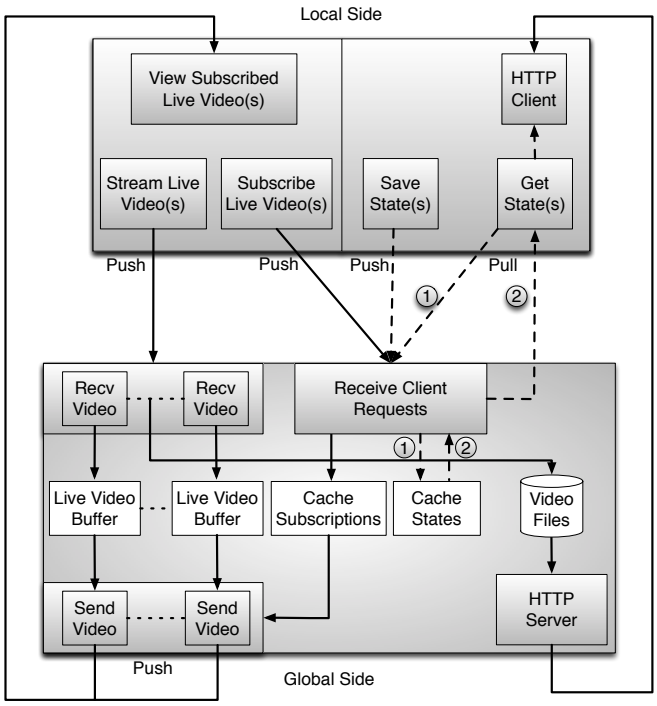

Fig. 3. The design of pVD.

III. Architecture

The architecture, shown in Figure 2, comprises functionality for handling in- and outgoing videos between a single user's computers. A subscription model for videos is used. To display a camera feed or a video at a computer, it must subscribe to the video. pVD allows a computer to send subscriptions to it. pVD will receive all subscriptions and use them to manage the switching of between computers.

A central part of the architecture is the distribution of live and stored videos to individual computers according to the subscriptions and the state of the videos. The functionality defined by the architecture includes streaming of outgoing videos, receiving incoming videos, buffering of video streams, and playback of videos.

The functionality of hand-over of videos between computers lets a computer save current video play back state, including where in the video play back is at, with pVD. When the video is streamed to a new computer the state information is used to let a user continue watching the video on the new computer from where it left off at the other computer.

The architecture allows two users to provide videos from and to each other's computers. User A will call user B through some channel (say, telephone or chat) and gets the address (present prototype uses the IP address) to the global side pVD of user B (call it pVD B). User A will indicate this address when starting a subscription to a video on one of user B's computers. The subscription handler of pVD A uses the address to contact pVD B and registers the subscription with itself as the receiver. When user B starts streaming a video, pVD B will look at its subscription data and send the stream onwards to pVD A. In turn, pVD A will forward the video in a normal fashion to user A's computer.

To aid privacy, a user must from the user interface of the computer with the video explicitly acknowledge the streaming of the video to another user's computer each time a streaming is started.

IV. Design and Implementation

The design of the system is shown in Figure 3. The system is separated into a local and a global side. The local side executes on each device. It comprises a user interface that sends requests to start and stop subscriptions, starts play-back of incoming live and stored videos from other computers, streams outgoing live video from cameras and sends stored videos. It also keeps track of the necessary state for handing over videos between computers.

The global side executes on one of the user's computers, typically a PC or laptop, with the necessary resources to serve or communicate with the other devices (bandwidth, enough storage and memory). It is assumed to be always on and accessible to the other computers.

The local side pushes videos, subscriptions and state data to the global side. The global side receives incoming videos and data, and pushes out video streams to computers with subscriptions. The global side manages information about subscriptions and the state of live and stored videos.

The local side is concurrent to the degree supported by the operating system running on the computer. Some smart phone operating systems may limit the concurrency possible. The global side is designed as a concurrent system executing on a general-purpose operating system. This is done to make it simpler and more flexible, and to be able to benefit with regards to performance from multiple cores.

Each frame in of a live video includes the sending computer ID, video ID, frame counter, and a time stamp for when the frame was captured. A subscription message indicates the user ID, computer ID of the viewer, and video ID. There are three state related messages to save, get and remove where (frame number) in a given video a specific user and computer is at.

Video files are served by a HTTP server at the global side. The system was implemented using Python and Python OpenCV. It runs on Linux and Mac OS X.

V. Evaluation

To characterize the performance of pVD, a set of experiments was conducted using ten computers. All computers were 2011-2012 Mac Minis at 2.7 GHz, 8 GB of memory, and connected by wire to the same 1 Gbit/sec Ethernet switch. Six computers were used to represent a user's computers having videos and cameras. These executed the local side. One computer was used to run the global side pVD. Three computers were used to represent local side viewers that subscribed to the produced video streams.

We measured the **subscribe round-trip latency**: the time it took from the local pVD sent a request to start a subscription until it was received and processed by the global side pVD and an acknowledgement was received back at the local pVD. To measure the subscribe round-trip latency, the subscribing computer records the time when it requests a subscription, and the time when an acknowledgement arrives, and calculates the delay. We increased the number of computers from one to six. Each computer sent one or ten subscription requests. The subscription experiment uses TCP/IP as the transport protocol.

We measured the **video end-to-end latency**: the delay from something happens in front of the camera at one computer until it is visible on the display at a subscribing computer. To measure the video end-to-end latency, we set up one local pVD computer with a camera capturing a user, and another local pVD computer subscribing to the camera and displaying the camera output onto a display. We arranged the user and the display such that a high frame rate video camera could record both on the same video. We recorded several videos of the user and the display. We then counted frames to see how many frames it took from the user initiated a movement until the movement became visible at the display.

We measured **resource usage** of the global pVD computer and the participating local pVD computers. Using the Python psutil module [16], we measured the CPU utilization at the global pVD computer, and the incoming and outgoing network traffic for both it and each of the other computers.

Videos were represented by point clouds from two Microsoft Kinect cameras per local pVD computer. These were sent using UDP messages, resulting in about 13.5 Mbits/sec per camera, or about 26.7 Mbits/sec of data from each local pVD computer. This is equivalent to about four HD videos (4 to 8 Mbits/sec) from each computer to the global pVD computer. In the results, we report the number of HD stream equivalents.

To simulate local pVD viewers, we used 3 Mac Minis as viewers, with each viewer receiving a copy of every stream sent to the global pVD computer. We gradually increased the number of camera computers from two to six, increasing the number of video streams to the global pVD computer and the number of outgoing streams from the global pVD computer.

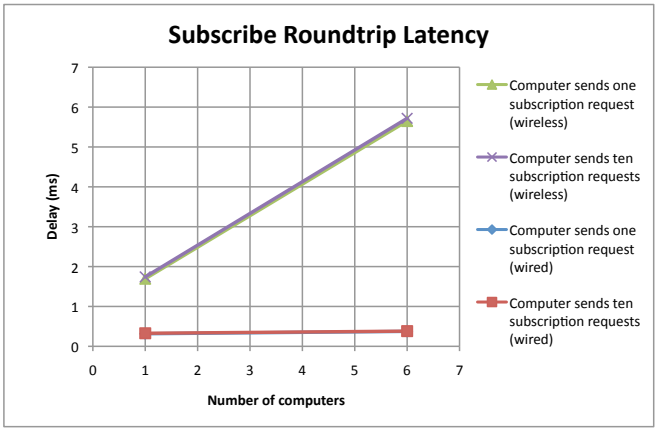

Fig. 4. Subscribe roundtrip latency on wireless and wired network. There is one subscriber per local pVD computer. Each subscriber sends one request in the first experiment and ten requests in the second experiment.

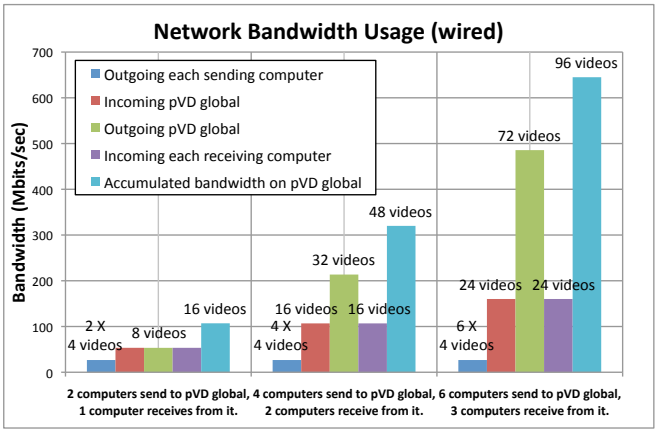

Fig. 5. Incoming and outgoing network bandwidth using wired connection.

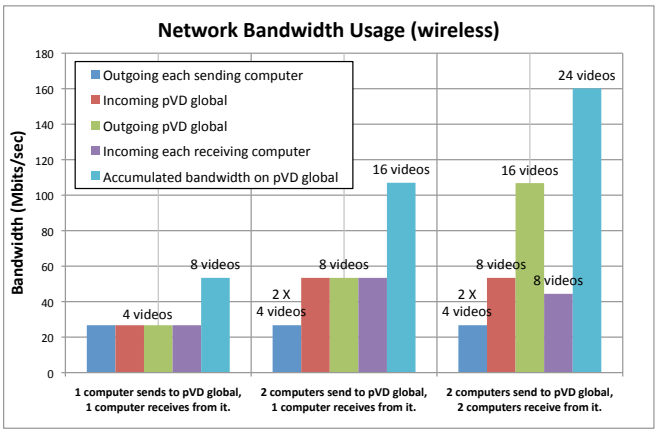

Fig. 6. Incoming and outgoing network bandwidth using wireless connection.

Figure 4 shows the subscribe round-trip latency. The round-trip latency is about 315 microseconds for one computer with one subscription request, and about 380 microseconds for six computers with ten subscription requests each. This is an insignificant increase. We conclude that the subscription

mechanism in the global pVD scales well with the number of computers and videos we expect a user to have.

The video end-to-end latency was between 90-125 ms. The variation in latency comes from several factors, including the distributed architecture of the prototype, the projector frame rate (120 fps), the video camera frame rate (240 fps), and that the Kinect (30 fps) can add 30 ms to the latency. We conclude that the video end-to-end latency is low enough to allow for interactive use.

In a study of latency [17], a 100 ms delay was noticeable by humans, but found acceptable. More than 200 ms delay made interaction uncomfortable. The sum of the subscribe message latency and the end-to-end latency is less than 200 ms. While at the borderline, the pVD system is able to stream live video events with latencies making it useful for interaction.

Figure 5 shows the network bandwidth usage at the computers involved and the number of HD stream equivalents in each experiment. With two local pVD computers, the global pVD receives four HD streams (26.7 Mbits/sec) from each computer resulting in a total of eight HD streams (53.4 Mbits/sec) on the network simultaneously. The local pVD viewer computer receives all of the eight video streams from the pVD global computer, resulting in a max load of sixteen HD videos in flight simultaneously on pVD global.

With six computers streaming to pVD global, we increased the number of pVD viewers to three. All of the viewers subscribe to every stream, so the global pVD sends out three times the incoming bandwidth (72 HD streams at 480 Mbits/sec). The total number of videos in flight simultaneously is 96 on pVD global. This pushes the system beyond an expected normal usage, but we have not observed any significant packet loss.

In summary, the accumulated bandwidth on pVD global with two senders and one viewer is 107 Mbits/sec, with four senders and two viewers is 320 Mbits/sec, and with six senders and three viewers is 645 Mbits/sec.

The CPU utilization on pVD global increases from 3.88 to 12.26% when it receives 8 to 24 videos and simultaneously sends 8 to 72 videos.

On a Gigabit network, the system can support in total 96 streams in the experiment. The CPU utilization is also less than 15% in this case. This is much more than the normal usage. We conclude that the results show the pVD global computer can easily be supported on even a low-end computer, and still have resources (like CPU or bandwidth) available for other applications and systems.

To characterize the impact a wireless network has upon pVD, we configured a system where the pVD global computer is connected by a wired 1 Gbit/sec Ethernet to an Apple Airport Extreme 802.11n (4th Generation) WiFi access point, and where the other computers use the WiFi network.

Figure 4 shows the subscribe roundtrip latency. The roundtrip latency is about 1.7 ms for one computer with one subscription request, and about 5.7 ms for six computers with ten subscription requests each. The video end-to-end latency was between 90-125 ms.

For the wireless configuration, figure 6 shows the network bandwidth usage and the number of HD stream equivalents in each experiment. With two computers streaming to and one computer receiving from the pVD global computer, eight videos were sent to and fully received at the receiving computer. The accumulated bandwidth at pVD global computer was 107 Mbits/sec. The receiving computer received 53.4Mbit/sec. When a second receiver was added, for a total of two receivers, each received only 44Mbit/sec instead of 53.4Mbit/sec. We believe the reduced bandwidth can be removed by a more modern wireless network with better performance and resistance to interference from other nearby wireless networks. However, the experiment shows that it is possible to wirelessly stream at least eight HD videos to the pVD global computer and to wirelessly receive at least eight HD videos from the pVD global computer.

VI. Discussion

pVD is based around a manual approach to controlling both video switching and privacy. A user must have access to all computers serving and consuming the videos. A user interacts directly with the pVD user interface on the sending and receiving computers. A user is the glue to bind together computers. When videos are sent between users a sending user must manually accept a one-time streaming of a video to another user. The sending user can at any time halt the streaming. This provides for some control of the privacy for the sending user. To strengthen the privacy we could have added techniques like time-outs for video streams, halting them automatically when the time-out occurs. However, this adds complexity, and we wanted to keep the pVD system as rudimentary as we could within reasons. The pVD system overall uses a simple and robust approach customized for a single user with a handful of computers. However, it does not extend and scale to many computers and to many users, and was not meant to do so.

pVD can do hand-over of video streams between computers, letting a video start playing again from where it was stopped at another computer. The state of all videos are stored at and handled by the pVD global side server, and not by the pVD local side computers. Because pVD is meant for a single user with just a handful of computers, there are no performance issues in doing this centralized. It also aids in doing hand-over of videos between computers by having the state of the videos at one place. However, if the pVD global looses state about the videos, the user must recreate it. We don't expect this to be an issue because of the usage domain with just a single user and a handful of computers.

The approach to share live and stored videos between multiple users will not scale to many users. It is intended to let a few users share live and stored videos in a case-by-case fashion. Users need to talk to each other to exchange enough information to connect. This can be automated and made more efficient, but we wanted to keep it basic and simple, and to involve the users in the sharing to reduce unintended sharing. While the multiple user approach demands users interaction, we still believe it is useful for simple ad hoc interaction and sharing between family and friends.

We have deliberately used a Gigabit wired Ethernet for some of the experiments where the goal was to measure the

performance behavior of the pVD global. A typical usage scenario is to have the pVD global computer connected by wire to a wireless access point and has a user's computers share videos with each other through a wireless network. A single 8 Mbits/sec HD video stream will in this set-up at the worst consume about 16 Mbits/sec of the wireless network. Wireless networks typically range from 54 Mbits/sec to 300 Mbits/sec. A wireless network should in practice be able to support a pVD configuration with a handful of computers and video streams. For the intended usage domain this is enough. Emerging wireless networks like the 802.11ac technology [18] can achieve 1300 Mbps and should together with future computers allow for even better performance for pVD.

VII. Conclusions

In this paper, we present a personal video distribution system for simple ad hoc sharing of live (camera) and stored video streams between a single user's computer.

The pVD system uses subscriptions to bind together a video and camera on one computer with the other computers. The user must have physical access to each computer to set up a subscription, and to start the streaming and receiving of a video. When a sender and a receiver run simultaneously, videos will start to flow according to the subscriptions. The system saves video state and allows the user to continue watching a video on another device, picking up from the same position in the video.

The system avoids issues with relying on third parties and access rights by being designed for very small scale and to be run only on a user's personal computers. The system can also be kept simple from not having to scale to support a large number of videos, computers and users.

The bandwidth is not a critical issue for the pVD prototype system and the system does not occupy too much system resources, like CPU. When all computers are on a Gigabit wired network, the system can support many simultaneous streams using a standard PC desktop for the pVD global side. The system also supports a handful of simultaneous streams on a wireless network. The system responses fast enough to user's requests and the latency for streaming live events will be noticed, but accepted by the user.

Acknowledgment

Many thanks to the technical staff at the department. This work was funded in part by the Norwegian Research Council, projects 187828, 159936/V30, 155550/420, and Tromsø Research Foundation (Tromsø Forskningsstiftelse).

References

[1] S. M. Habib, S. Ries, and M. Muhlhauser, "Cloud computing landscape and research challenges regarding trust and reputation," in *Ubiquitous Intelligence & Computing and 7th International Conference on Autonomic & Trusted Computing (UIC/ATC), 2010 7th International Conference on*. IEEE, 2010, pp. 410–415.

[2] I. Ion, N. Sachdeva, P. Kumaraguru, and S. Čapkun, "Home is safer than the cloud!: privacy concerns for consumer cloud storage," in *Proceedings of the Seventh Symposium on Usable Privacy and Security*. ACM, 2011, p. 13.

[3] E. Chin, A. P. Felt, V. Sekar, and D. Wagner, "Measuring user confidence in smartphone security and privacy," in *Proceedings of the Eighth Symposium on Usable Privacy and Security*. ACM, 2012, p. 1.

[4] [Online]. Available: http://www.pps.tv/

[5] [Online]. Available: http://www.pptv.com/

[6] [Online]. Available: http://www.livecast.com/

[7] [Online]. Available: http://qik.com/

[8] [Online]. Available: http://www.upnp.org/

[9] [Online]. Available: http://www.dlna.org/

[10] C. Ullrich, R. Shen, R. Tong, and X. Tan, "A mobile live video learning system for large-scale learning—system design and evaluation," *Learning Technologies, IEEE Transactions on*, vol. 3, no. 1, pp. 6–17, 2010.

[11] K. Wolf, S. Linckels, and C. Meinel, "Teleteaching anywhere solution kit(tele-task) goes mobile," in *User Services Conference: Proceedings of the 35 th annual ACM SIGUCCS conference on User services*, vol. 7, no. 10, 2007, pp. 366–371.

[12] H.-Y. Chang, Y.-Y. Shih, and Y.-W. Lin, "Cloudpp: A novel cloud-based p2p live video streaming platform with svc technology," in *Computing Technology and Information Management (ICCM), 2012 8th International Conference on*, vol. 1. IEEE, 2012, pp. 64–68.

[13] F. V. Hecht, T. Bocek, R. G. Clegg, R. Landa, D. Hausheer, and B. Stiller, "Liveshift: Mesh-pull live and time-shifted p2p video streaming," in *Local Computer Networks (LCN), 2011 IEEE 36th Conference on*. IEEE, 2011, pp. 315–323.

[14] E. Kim and C. Lee, "An on-demand tv service architecture for networked home appliances," *Communications Magazine, IEEE*, vol. 46, no. 12, pp. 56–63, 2008.

[15] A. Kaheel, M. El-Saban, M. Refaat, and M. Ezz, "Mobicast: a system for collaborative event casting using mobile phones," in *Proceedings of the 8th International Conference on Mobile and Ubiquitous Multimedia*. ACM, 2009, p. 7.

[16] [Online]. Available: http://code.google.com/p/psutil/

[17] A. Pavlovych and W. Stuerzlinger, "Target following performance in the presence of latency, jitter, and signal dropouts," in *Proceedings of Graphics Interface 2011*. Canadian Human-Computer Communications Society, 2011, pp. 33–40.

[18] [Online]. Available: http://www.apple.com/airport-extreme/