MultiStage: Acting Across Distance

Fei Su, Giacomo Tartari, John Markus Bjørndalen,

Phuong Hoai Ha, Otto J. Anshus

Department of Computer Science
University of Tromsø, Norway
{fei.su,giacomo.tartari}@uit.no, {jmb,phuong,otto}@cs.uit.no

Abstract. We report on a prototype system helping actors on a stage to interact and perform with actors on other stages as if they were on the same stage. At each stage four 3D cameras tiled back to back for an almost 360 degree view, continuously record actors. The system processes the recorded data on-the-fly to discover actions by actors that it should react to, and it streams data about actors and their actions to remote stages where each actor is represented by a remote presence, a visualization of the actor. When the remote presences lag behind too much because of network and processing delays, the system applies various techniques to hide this, including switching rapidly to a pre-recorded video or animations of individual actors. The system amplifies actors' actions by adding text and animations to the remote presences to better carry the meaning of actions across distance. The system currently scales across the Internet with good performance to three stages, and comprises in total 15 computers, 12 cameras, and several projectors.
Keywords: Temporal Synchronization; Remote Interaction; Computer Mediated Collaboration.

1 Introduction

We envision computer mediated collaborative performances where actors at physically remote locations, as illustrated in Figure 1, interact and coordinate their actions as if they are next to each other on the same stage or in the same room. Through various means, including audio, video and animations, each actor has a remote presence at one or several remote stages. We are interested in how to mask the effects of delays and distance.

In this paper we describe a system doing this for the visual side of a remote presence: MultiStage collects state, like video, about each stage through various sensors, like cameras and microphones, and analyses the observed state to identify information like actor gestures. State data and information is streamed between stages to maintain a remote presence for each actor, and to monitor and control the system.

Each stage has several incoming data streams that are used to create a presence of remote actors. Actors in a room watch and react to the remote presence of other actors. There can also be several third parties, audiences, just observing,

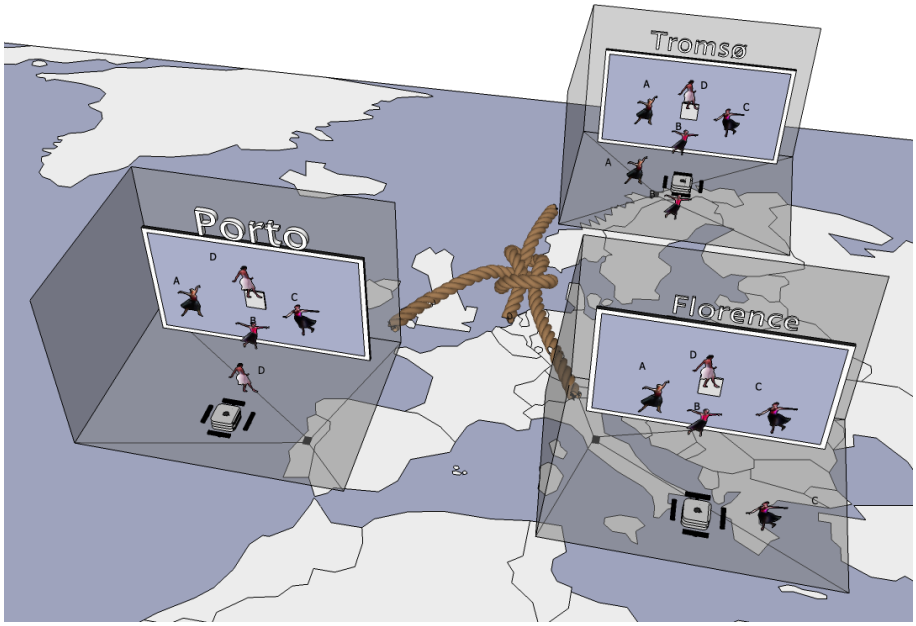

Fig. 1. Four dancers at three different stages dance together. Each stage is equipped with sensors to detect actors and a display to visualize the remote presence of all the performers. The rope and knot represent the global system binding together the stages.

and not directly participating. Audiences can be physically present at one of the stages, or be on the Internet. An audience local to a stage can watch the local physical events unfolding, and watch visualizations of both the local and remote events.

In principle, there will always be some delay from an event happens until it can be observed. Light alone needs 134ms to travel the length of Earth's equator. In practice, the total delay when observing a remote event includes delays coming from the sensors, transferal of data from sensors to computers, processing of the sensor input, network transmission, on-route processing, receiving and processing the received data, and preparing and visualizing the data locally. Even if the delays can be reduced, they can never be removed. Consequently, we have to live with the delays, and find ways of reducing the effect they have on the actors and the audiences. The effect of the delays can be reduced through different techniques including on-the-fly manipulation of the remote presence representation of actors. We must also mask the effect of distance. On a theater stage the actors use several techniques including costumes, makeup, and exaggerated movements to reach out to the audience. We propose to let a user instruct the system through gestures to add enhancements to the remote presence. For example, a given arm movement could be turned into a text bubble

above the visualization of the user, or a glowing halo around the arm. We call this *amplified interaction*.

There are many commercial tele-conferencing and messaging systems where two or several persons interact through instant text, video and audio as well as file transfer. The latencies can be quite tolerable. However, teleconferencing systems are best when used in unstructured interaction without interactively fast synchronized movements of participants. Tele-conferencing systems are typically not flexible with regards to manipulating remote presences, and how they are arranged on, say, a display. They also lack functionalities for amplified interaction.

2 Related Literature

Several research systems for collaboration exist. In [1], [2] and [3] two room interaction systems are described with a focus on achieving audio synchrony. They compensate for the network latency by delaying local actions correspondingly, making both rooms experience the same delay. In [2] a series of experiments based on the DIP system is described with focus on the audio delay, and how the delay affects musician's cooperation. An artificial delay of 50ms to the remote room's audio stream was tolerable. With the same latency added at both rooms it became possible to play easily together with a delay of up to 65ms. This approach used by DIP for audio can also be used by MultiStage for video.

In [4] a remote camera system for teleconferencing supporting user cooperation between a local and a remote room is described. The system captures 360-degree images as well as supports pan/tilt/zoom of cameras. The audio and video can be recorded. Consumed network bandwidth is from 1.95 to 7.4MBytes/s.

In [5] a three-room distributed collaboration system is described, allowing three people to collaborate in a virtual environment. At each room there is a multi-touch table, camera, speaker, microphone, and two LCD monitors to display the two other rooms. The shadow of remote hand and arm gestures are captured by an infrared camera and displayed on the multi-touch table to show the remote person's behavior.

In [6] a remote presence system using a remote controlled android is described. The state of the android includes idle, speaking, listening, left-looking and right-looking. A teleoperator control android's behavior by choosing its state. They conclude that using an android gives a strong remote presence.

In [7] a system intended for informal meetings between rooms is described. The system merges the images from panorama cameras acquiring the background of a room, with a camera acquiring the users when they are close by the display. The system amplifies the remote presence of the users by allowing users to maintain eye contact during a conversation.

In [8] a multi-camera real-time 3D modeling system for tele-presence and remote collaboration is described. 3D models of users are computed from 2D images from multiple cameras, and the 3D models are streamed to remote rooms

where users are visualized in a virtual 3D environment. Computing and visualizing collisions and reaction forces to virtual objects in the virtual space strengthen the remote presence. The system is built on top of a middleware that simplifies the use of a compute cluster to obtain 3D meshes and textures from the cameras.

In [9] a multi-modal corpus for research into human to human interaction through a virtual environment is presented. The virtual environment is defined as a virtual dance studio where a dance teacher can teach students choreographies. Both teacher and students are represented in the virtual studio by 3D avatars. The corpus consists of the recordings of the 3D avatars and outputs form other sensors, such as cameras, depth sensors, audio rigs and wearable inertial measurement devices. A dance instructor and a musician provided also some ground truth annotations for the corpus.

In [10] a study on hand gesture speed classification with the goal to improve the human-computer interaction is presented. The aim of the study is to train a virtual human to detect hand's movement in a noisy environment. The factors of the study are multiple body features like hand, wrist, elbow and shoulder, evaluated against different gesture speed such as slow, normal and fast.

3 Temporal Causal Synchrony between Actors

Some actions by actors are causally related. One actor does an action, and some time later another actor does an action because of the first action. A system must preserve the order of the actions when they are causally related.

Even if causality is preserved, there is a delay between an action and the corresponding reaction(s), and the system should ideally keep the delay low enough to make actors experience it as if they would when on the same physical stage. How large the delay is indicates how well actors are in temporal causal synchrony.

We define actors to be in *loose* temporal causal synchrony with each other when there are no special demands on delays. This is typically the case in unstructured interaction where it does not matter a great deal if actions by actors are slightly delayed or out of order with each other. This will typically be the case in teleconferencing with approaches like Skype.

However, for structured interaction with coordinated movements, as in synchronized dancing and in rapid action-reaction situations like, say, martial arts, correct causal ordering and short delays become critical to preserve the illusion that the actors are on the same stage. We define *interactive* temporal causal synchrony to be when actions by an actor is seen in causal order and as fast as actors are used to when being on the same stage.

Delays are unavoidable, and they can be large and even varying enough so that interactive temporal causal synchrony can not be achieved. In these cases we must mask the effects of the delays to create an illusion of synchrony. Some approaches are outlined in the following.

Actor feedbacks: The actors reacts to the remote presence videos as if they were the actual other actors. Depending on how large the delays are and

how much they vary, the interactions can become awkward. Only loose temporal causal synchrony can be expected to be achieved.

Shared clock, shared performance start-time, individual actor scripts: We synchronize the clocks of all computers, set a performance start-time and begin a count-down at each stage. When the count-down finishes each actor starts acting according to a script defining what the actor should do and when the actor should do it (for instance, two actors doing handshake). Assuming that the scripts are made correctly, even if the actors don't actually interact it will seem to an audience as if they do. In this approach the scripts have been made with knowledge about the delays, and each script tell the actor when, modified by the delay, to do an action.

Shared clock, individual performance start-time, individual or shared actor scripts: We synchronize the clocks, and select a start-time for the performance. We select one stage to be the live stage. The other stages are secondary stages. We measure the delay from the live stage to all secondary stages and modify the start time of each stage's count-down according to the delay between it and the live stage. When the count-downs finishes, all remote presences will move at the same time and in synchrony with the actors present at the live stage. The actors and the audience at the live stage will see the other stages as if they are in interactive temporal synchrony with the live stage. Actors and the audiences at the secondary stages will experience the effects of delays.

Act-by-wire: We synchronize the clocks, and start the stages at the same time. The computers are continuously monitoring and measuring several metrics including delays between stages. If one or several videos are arriving late because of delays, the computers do on-demand manipulations and animations of the live video or substitute the live video with a pre-recorded video. If there are scripts available telling the system what each actor was meant to do, the system can create animations mimicking the expected movements. If there are no scripts telling the system what to do, it can try to predict the movements of an actor based on the most recent movements, or it can blur what is going on such that the audience not so easily notices the delays. In all cases, the goal is to create an illusion of interactive temporal causal synchrony.

4 Amplified Actor Interaction and Gestures

On a theater stage, with a significant physical distance between actors and the audience, bold makeup, clothes, and exaggerated movements are used to better project to the audience what the actors are doing.

In remote interactive performances there is a distance not only to an audience, but also between the actors. Consequently, the actors need their appearance, movement and gestures to be amplified such that they become easier to see and understand both for the other users and for the audience. In this way we extend the range of human interaction to remote locations and enrich the communication between them. We term this amplified interaction.

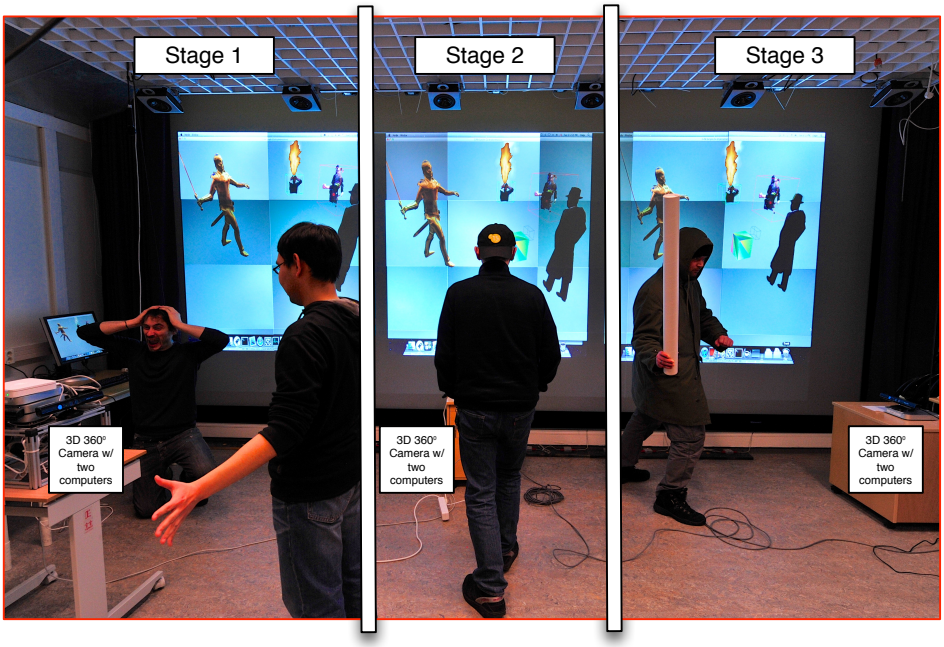

Fig. 2. To do experiments, MultiStage is set up with three stages and four actors in the same room. Each stage has its own camera rig. Each stage displays all actors. The global system binding together the stages are located either locally or on a remote computer across the Internet. Note: the flame animation has been enhanced in the figure for better visibility.

To be able to detect what an actor is doing, we must surround him with an interaction space [11]. An interaction space detects human movements, and analyzes them looking for gestures. A gesture represents a pre-defined command to the system to execute code to do some functionality.

A gesture can be simple, like raising an arm, or complicated like doing two-arm movements. They can also be active like walking in a specific direction or passive as in standing still posturing. A collective (collaborative) gesture is a combination of the above kinds of gestures. Collective gestures can happen at the same stage, or be distributed, comprised of gestures from multiple stages. For example, when two actors at different stages, within some short timespan, raise their left arms above their head this can be interpreted as, say, a command to the system to animate a lightning between the two raised arms and display it on all the displays.

Based on the gestures we can create effects in the remote presence manifesting itself at remote rooms. A user's arm movement can in the remote presence be amplified by having a text bubble appear in the video, and by adding other visual effects to the representation of the user. The users remote presence can

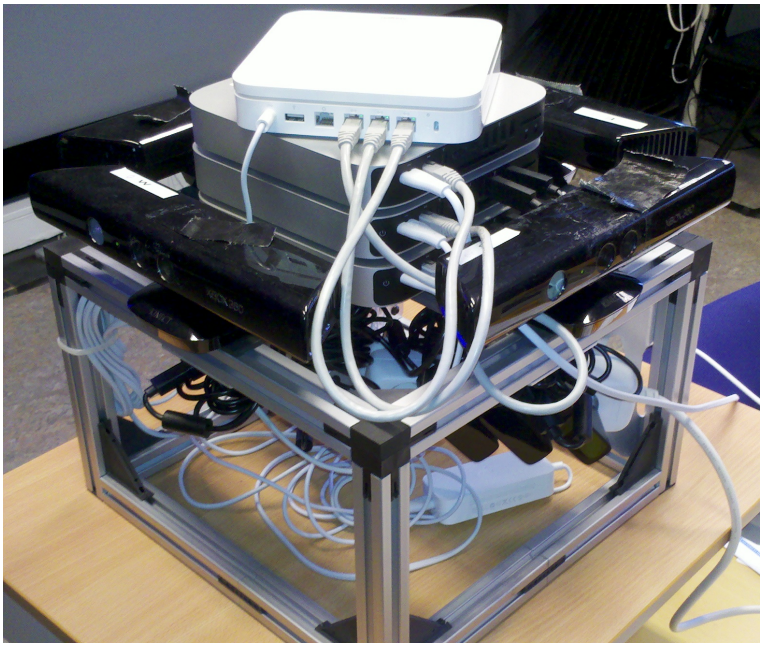

Fig. 3. The four 3D Kinect camera rig used at each stage for almost 360 degrees coverage.

even be enhanced by executing a model of the user and using its output as the basis for the remote presence.

To experiment with the system, we set up three stages, named stage 1, 2 and 3, see figure 2, in a single room. There are two actors on stage 1, and one actor at each of the other two stages. Even if all three stages were co-located in the same room they each occupied a different area of the room, and they each had their own interaction space and display. Each interaction space uses four Kinect 3D cameras, see figure 3. The cameras are arranged in a square with two computers receiving camera output and doing processing on the images. Four Kinects arranged in a square cover almost 360 degrees. We typically place the camera rig in the middle of a stage, and act around it. The room where the stages are located has a large 6m by 3m display wall. Each stage displays the remote presences of local and remote actors onto its assigned area of the display wall.

To simulate both the situation when all stages are on the same local network as well as when they are connected through a wide area network, the Internet, we locate the global side handling the distribution of data between the stages either locally at Tromsø or at a computer in Oslo or Copenhagen.

The images picked up by the cameras are analyzed and sent as data streams to all stages. This data represents the actors and to some degree what they are doing. The data is used to create a remote presence of each actors. This can take

the form of a simple video, a manipulated video, or an animation of the actor as illustrated in the figure. Each stage has a display where the remote presence of each actors is displayed inside the same virtual stage.

On the virtual stage three of the actors have been amplified. At Stage 1 the kneeling actor with hands on his head is interpreted by the system as showing agitation, and the system has added an animated fire above his remote presence. The other actor at Stage 1 does nothing the system recognizes, and a low resolution video of him is displayed at all stages. The actor at Stage 2 knows that if he keeps his hands in the pocket, has a hat on, and emulates walking, his remote presence will be that of an animated figure of a walking man with long dark coat and a hat. The actor at Stage 3 knows that if he has something looking like a sword in his right hand his remote presence will be that of a knight with a sword.

Presently, the prototype system cannot do all of the described functionality. The actual dynamic gesture and posture recognition is not yet in place. Consequently, the three amplified remote presences in Figure 2 were predetermined to be what they are.

5 Design and Implementation of Prototype

The design of the prototype, please see Figure 4, comprises several systems including the collaboration system, the human interaction system, the administrator interaction system, and an internal state & performance monitoring system.

The MultiStage system has a local side and a global side. The local side primarily focuses on what is happening locally on a stage. The systems implementing the local side executes on computers local to a stage. These systems include:

(i) the local detection system doing local state monitoring (LSM) recording what the cameras see, and doing on-the-fly local analysis (LSA) of the data to find interesting objects and events in the videos. The data is streamed to the global side for further analysis and distribution to the other stages.

(ii) the remote presence system subscribing to data streams from the other stages, and creating a remote presence of remote actors. Presently the primary remote presence technique is to visualize remote actors on a very large display per stage. In the future we may add physical devices like robots to the remote presence.

(iii) the human interaction system inform actors on when they should start actions, like moving arms, according to a given script. It will also in the future enable an actor to give gesture input to control a remote physical presence, like a robot and manipulating how the actor is displayed.

(iv) the temporal causal synchrony system applies the techniques discussed previously in this paper to reduce the effects of delays.

The global side is the glue binding the stages together, taking care of distribution of data between stages, and doing analytics needing data from multiple stages. The global side includes these systems:

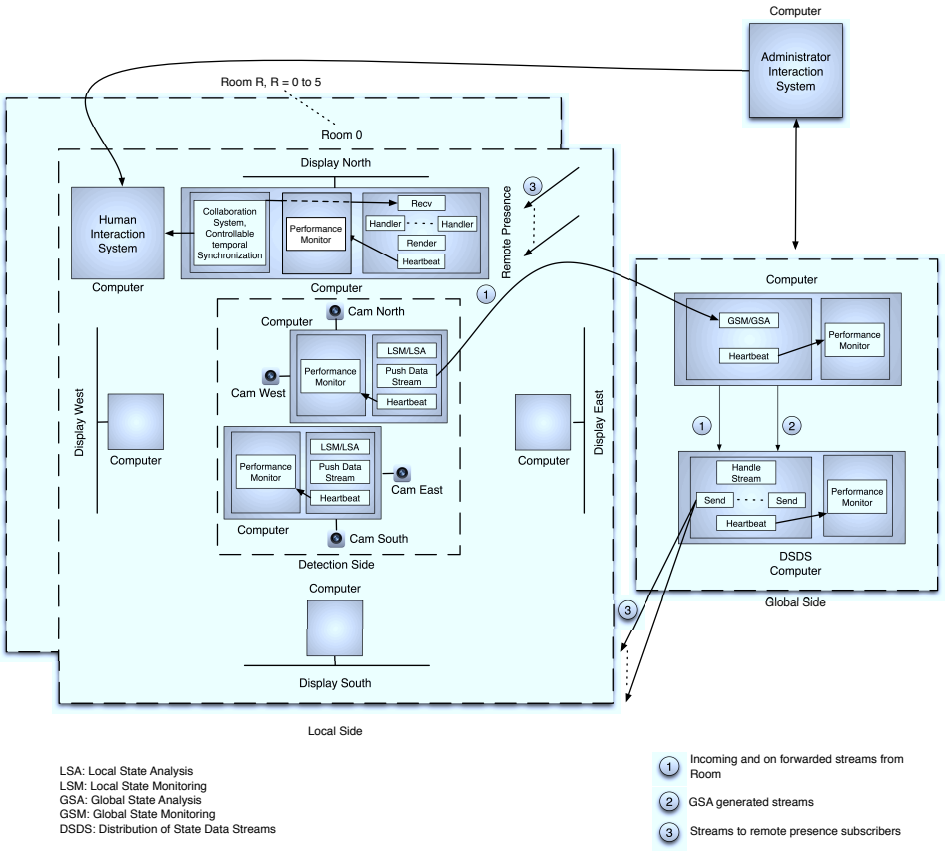

Fig. 4. The Design of MultiStage showing the systems at each stage and the global systems binding stages together.

(i) the administrator (or director) interaction system lets an administrator/director manage the systems, and setting start times for performances.

(ii) the global state detection system doing global state monitoring (GSM) collecting data from all the stages, and making it available for on-the-fly global state analysis (GSA) to detect distributed state like collective gestures and collisions when actors at different physical stages occupy the same volume on the virtual stage.

(iii) the distribution of state data streams (DSDS) system managing subscriptions from stages for data streams, and doing the actual transmitting of data to the remote presence computers locally to the stages.

Both the local and global side executes the internal state and performance monitoring system doing live performance measurements of several metrics in-

cluding latency and bandwidth. These are made available to the global sides administrator interaction system. The performance measurements are also made available to the temporal causal synchrony system.

The systems were implemented on the operating systems Linux and Mac OS X and using several languages including C, Python and the Go programming language [12]. The animations and 3D models are rendered using the Horde3D graphical engine [13].

The prototype in Figure 2 can be configured to run on a variable number of computers. We typically have three to four computers per stage, two for the global side, and one computer for the administrator interaction system. With three stages the prototype comprises in total 12-15 computers. All computers can be connected through a combination of wireless network, switched gigabit Ethernet network and a wide area network (between Tromsø and Oslo (1500km)).

6 Evaluation

To characterize the performance of MultiStage a set of experiments were conducted. All computers used were modern Mac Minis at 2.7GHz. Each stage had three computers: two with two cameras each, and one with a large display. The global side had two computers: one for the global state monitoring and analysis, and one for the distribution of state data streams. Each stage and the global side had a network switch each. All switches were connected to a switch with access to the Internet.

For all experiments all stages were on the same 1GHz switched Ethernet LAN inside the Department of Computer Science at the University of Tromsø. The DSDS, the system distributing data streams to the stages, was either on the same LAN as the stages, or located on a Planetlab [14] computer at the University of Oslo, 1500km away. In this case, all data sent between stages went from Tromsø to Oslo and back again. This separates the stages across the Internet.

Using the Python Psutil module [15], we measured the CPU utilization, amount of physical memory in use, and incoming and outgoing network traffic for all computers in use. We also measured three types of latencies: (i) the latency between the global side DSDS computer and the stages. We measured this by recording the time when we send a message from DSDS to a stage, and recording when a reply message comes back to DSDS; (ii) the end to end latency: the time it takes for a physical event happening on a stage to be picked up by the cameras and until a visualization of the actor is actually displayed on the same stage. We used a video camera with a high frame rate to record several videos of a user and the remote presence done on a display behind the user. We then counted frames to see how many frames it took from the user moved to the visualization caught up; (iii) the latency an actor can tolerate before the illusion of being on the same stage breaks. We subjectively decided this through two experiments. In the first we had an actor moving his arms while we observed him and his remote presence simultaneously. In software we artificially added

a delay to the remote presence until we subjectively decided that the remote presence lagged too much behind to be mistaken for being on the same stage. In the second experiment an actor shaked hands with a remote actor. The delay between the actors was artificially increased until we subjectively decided that the handshake was not happening as fast as it would if the actors were physically on the same stage.

Factors in the experiments were the number of stages (1 to 3), the resolution of the images from the cameras (bounding box alone, 1000 to 5000 points per image), the number of cameras per stage (0 to 4), and the location of the DSDS subsystem distributing data between stages (LAN in Tromsø vs. WAN to Oslo).

The results show that the resource usage in all cases are either very low or low. The implication is that the system is not resource limited. There is practically no loss of data in the experiments with the DSDS on the same LAN as the stages. When we separate the stages with a WAN by locating the DSDS on a computer in Oslo 1500km away, we see just an insignificant increase in data not getting across to all stages. The implication is that we can expect that the system typically will have satisfactory bandwidth available even when the stages are separated by the Internet.

When all stages and the global side were on the same LAN, the round-trip latencies were between 1-2ms. When the DSDS system was on a computer in Oslo the round-trip latencies were around 32ms. This matches well with measurements reported by PingER [16] for Europe.

On a LAN the end to end latency was between 90-125ms. With the DSDS at the computer in Oslo, the end to end latency was between 100-158ms. Two times the end to end latency, 200-316ms, is the delay that actors will experience from they do an action until they see a visualization of another actor reacting. We term this the **actor to actor latency**.

We subjectively decided that movements being delayed less than 100ms maintains the illusion of being on the same stage. However, the objective measurements show that an actor to actor latency is at typically 300ms. Consequently, the system should apply its techniques to mask the effects of the too long delay.

In the handshake experiment, we decided that an actor to actor latency of about 600ms was just acceptable and could be mistaken for how people shake hands when both are present in the same room. Longer delays bordered on creating a feeling that the remote actor was being obnoxious by delaying just a bit too long before responding to a hand shake. This indicates that the prototype is able to maintain the illusion of being on the same stage for hand-shake type of interactions.

The variation in latency we measured is because of several factors, including the distributed architecture of the prototype and the frame rate of the projector, video camera (240 fps) and the Kinects (30 fps), and other traffic on the LANs and WAN.

7 Conclusions

The subsystems and bindings between them makes for a complex actor collaboration system. While good programming practices will reduce the number of failures, a simpler system will provide for a higher probability of avoiding failures right before and during a performance. We will simplify based on the lessons learned from the prototype.

We believe that the built-in on-line monitoring of the state of the individual components of the system is important to discover where problems happen, and to help in fixing them. The on-line performance monitoring is critical for discovering delays long enough so that the system can try to mask their effect.

Having stages across the Internet is a challenge for the system because traffic load, failures and outages are mostly unknown before they happen. We have documented that the system scales to at least three stages with a total of at least 12 outgoing and 36 incoming data streams. Based on the performance measurements we conclude that the location of the data stream distribution server binding together the stages is not critical for the end to end latency of the system when it is used to do natural interaction, like handshakes, where delays of even 600ms is tolerable. However, when movements are meant to happen simultaneously and synchronized, the distribution server should be located where it provides for the lowest latencies. Data available in services like PingER [16] can help to choose a location to minimize latency between stages.

With regards to bandwidth, the location of the distribution server is presently not critical. This may change if the data streams grow in size and number. However, if the global analyzer and distribution sub-systems are located on computers on the same local area network as one or more of the stages, the Internet traffic is significantly reduced. This will penalize the other stages but could be useful for a performance with local audiences or where synchronized interactions are mostly among actors on the local stages.

Even if the system can do temporal synchrony and mask away delays, it is not yet clear how practical the system is in actual use. While we have not done formal user study experiments exploring the system capabilities with actors needing to tightly coordinate their movements, we have documented the performance limits of the MultiStage system. This provides for a sound prototype platform for experiments in a context of distributed performances with real actors.

Acknowledgment

We would like to thank Ken Arne Jensen for helping us build the camera rigs and make it look like something done by the creatures in Alien, Jon Ivar Kristiansen for help with the network and silently testing us by giving us a broken switch, and Maria Wulff-Hauglann for being brave enough to lend us shiny new Mac Minis so we could do a third stage.

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. Sawchuk, A., Chew, E., Zimmermann, R., Papadopoulos, C., Kyriakakis, C.: From remote media immersion to distributed immersive performance. In: Proceedings of the 2003 ACM SIGMM workshop on Experiential telepresence, ACM (2003) 110–120
2. Chew, E., Kyriakakis, C., Papadopoulos, C., Sawchuk, A., Zimmermann, R.: Distributed immersive performance: Enabling technologies for and analyses of remote performance and collaboration, NIME 06 (2006)
3. Zimmermann, R., Chew, E., Ay, S., Pawar, M.: Distributed musical performances: Architecture and stream management. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP) **4**(2) (2008) 14
4. Sato, Y., Hashimoto, K., Shibata, Y.: A new remote camera work system for teleconference using a combination of omni-directional and network controlled cameras. In: Advanced Information Networking and Applications, 2008. AINA 2008. 22nd International Conference on, IEEE (2008) 502–508
5. Tang, A., Pahud, M., Inkpen, K., Benko, H., Tang, J., Buxton, B.: Three's company: understanding communication channels in three-way distributed collaboration. In: Proceedings of the 2010 ACM conference on Computer supported cooperative work, ACM (2010) 271–280
6. Sakamoto, D., Kanda, T., Ono, T., Ishiguro, H., Hagita, N.: Android as a telecommunication medium with a human-like presence. In: Human-Robot Interaction (HRI), 2007 2nd ACM/IEEE International Conference on, ACM (2007) 193–200
7. Dou, M., Shi, Y., Frahm, J., Fuchs, H., Mauchly, B., Marathe, M.: Room-sized informal telepresence system. In: Virtual Reality Workshops (VR), 2012 IEEE, IEEE (2012) 15–18
8. Petit, B., Lesage, J., Menier, C., Allard, J., Franco, J., Raffin, B., Boyer, E., Faure, F.: Multicamera real-time 3d modeling for telepresence and remote collaboration. International journal of digital multimedia broadcasting **2010** (2009)
9. Essid, S., Lin, X., Gowing, M., Kordelas, G., Aksay, A., Kelly, P., Fillon, T., Zhang, Q., Dielmann, A., Kitanovski, V., et al.: A multi-modal dance corpus for research into interaction between humans in virtual environments. Journal on Multimodal User Interfaces (2012) 1–14
10. Elgendi, M., Picon, F., Magnenat-Thalmann, N.: Real-time speed detection of hand gesture using kinect. In Springer, ed.: Proceedings of the Autonomous Social Robots and Virtual Humans workshop, 25th Annual Conference on Computer Animation and Social Agents. (2012)
11. Stodle, D., Troyanskaya, O., Li, K., Anshus, O.: Tech-note: Device-free interaction spaces. In: 3D User Interfaces, 2009. 3DUI 2009. IEEE Symposium on, IEEE (2009) 39–42
12. Go: http://golang.org/
13. Horde3d: http://www.horde3d.org/
14. Planetlab: https://www.planet-lab.eu/
15. Psutil: http://code.google.com/p/psutil/
16. PingER: http://www-iepm.slac.stanford.edu/pinger/