electronics

MDPI

Article

Low Delay Video Streaming on the Internet of Things Using Raspberry Pi

Ulf Jennehag *, Stefan Forsstrom * and Federico V. Fiordigigli

Department of Information and Communication Systems, Mid Sweden University, SE-85170 Sundsvall, Sweden; fedfio@student.miun.se

* Correspondence: ulf.jennehag@miun.se (U.J.); stefan.forsstrom@miun.se (S.F.); Tel.: +46-10-142-8745 (U.J.); +46-10-142-8574 (S.F.)

Academic Editors: Simon J. Cox and Steven J. Johnston
Received: 29 April 2016; Accepted: 13 September 2016; Published: 20 September 2016

Abstract: The Internet of Things is predicted to consist of over 50 billion devices aiming to solve problems in most areas of our digital society. A large part of the data communicated is expected to consist of various multimedia contents, such as live audio and video. This article presents a solution for the communication of high definition video in low-delay scenarios (<200 ms) under the constraints of devices with limited hardware resources, such as the Raspberry Pi. We verify that it is possible to enable low delay video streaming between Raspberry Pi devices using a distributed Internet of Things system called the SensibleThings platform. Specifically, our implementation transfers a 6 Mbps H.264 video stream of 1280 × 720 pixels at 25 frames per second between devices with a total delay of 181 ms on the public Internet, of which the overhead of the distributed Internet of Things communication platform only accounts for 18 ms of this delay. We have found that the most significant bottleneck of video transfer on limited Internet of Things devices is the video coding and not the distributed communication platform, since the video coding accounts for 90% of the total delay.

Keywords: Raspberry Pi; Internet of Things; video; streaming; low delay

1. Introduction

The number of smart electronic devices, such as smartphones, different wearables, and connected appliances, has increased significantly. A network of electronic devices like these, capable of communicating with each other to reach common goals, can be referred to as the Internet of Things (IoT) [1]. The devices are able to observe and interact with the physical environment, which allows the IoT to influence our lives significantly via applications in home automation, security, automated devices, health monitoring, and management of daily tasks. Current estimations claim that there will be over 50 billion connected devices as soon as 2020 [2], of which many will be typical IoT devices, such as small embedded computers (e.g., Raspberry Pi devices) or different wireless sensor networks. It is expected that the majority of the data traffic generated from these devices will be multimedia data, and this multimedia traffic will account for 80% of all Internet Protocol (IP) traffic by 2019 [3]. Some even claim that multimedia is such an essential part of IoT that a new paradigm has been suggested: "the Internet of Multimedia Things" [4].

There are many articles that research IoT problems in the area of identification, sensing, communication technologies, security, and multimedia streaming. The area of multimedia communication in the IoT for low end-to-end delay video streaming in time critical applications using resource constrained hardware is however little explored. Therefore, this article focuses on the problem of communicating high definition live video for IoT applications in surveillance scenarios with low delay under the constraints of typical cheap IoT devices such as the Raspberry Pi. A particular scenario

under consideration is the temporary surveillance deployment for a construction site, where live surveillance of equipment and personal safety is required. Another interesting scenario is surveillance of different types of events, such as concerts or festivals. The novelty of this research is not in the scenario itself, but in the results of our investigation regarding whether or not it is possible to communicate high definition live video on the IoT with low delay under the constraints of typical cheap IoT enabled devices. This article seeks to find a system that meets the following requirements:

1. A low delay from source to sink of $<$200 ms, providing a video viewing experience as close to live as possible.
2. High definition video content of 1280 $\times$ 720 at 25 frames per second (FPS), to be able to make out details in the captured video.
3. Runs on a cheap resource constrained device such as a Raspberry Pi, to show that the solution will be viable for the IoT.

We expect our results to show that it is possible to enable low delay multimedia IoT applications using distributed systems techniques under the constraints of typical IoT devices with limited hardware capabilities. The remainder of this article is organized as follows: Section 2 presents related work and our approach to meet the stated requirements. Section 3 presents our results, verification, and measurements. Finally, Section 4 presents our final discussion and conclusion.

2. Materials and Methods

Our approach is based on the idea of using distributed systems techniques to enable real-time video streaming on the IoT. In particular, it combines peer-to-peer (P2P) technology with Distributed Hash Tables (DHT) to enable scalable communication with low delay, in order to send minimal chunks of encoded video as small P2P packets to minimize delay of the video transmission. This section will be split into two parts. The first part will provide information about related work and background theory of our work. The second part will present our method and detailed approach to meet the requirements and solve the problem.

2.1. Related Work

The related work presented in this section will provide an understanding of the state-of-the-art in low delay video streaming over the Internet, and the IoT communication systems that are currently employed on the IoT.

2.1.1. Low Delay Video Streaming

There is much related work in the area of multimedia transfer and in particular surveillance applications. Jiang et al. analyse current research in real-time data exchange and propose a platform and a Control over UDP (CoUDP) protocol for performing multimedia transmission on the IoT [5]. Their system is, however, built on several centralized components which can add unnecessary overhead when it comes to minimizing delay. Martinez et al. study the performance of Dynamic Adaptive Streaming over HTTP (DASH) when it streams a video over a Content Centric Networking (CCN) architecture in typical IoT scenarios [6]. Since the system is built on CCN, it is not particulary feasible in real world scenarios where almost all networks are IP-based. Similar work related to surveillance on the IoT using resource constrained devices can be found in [7–10]; however, none of these focus specifically on low latency video and they do not present any measurements on how low delay they achieved in their applications.

Multimedia data represents the majority of Internet traffic today. As a result, there is an increased demand for high resolution video. In order to not saturate the connection bandwidth and to achieve a real-time transmission, adequate video compression is required. One of the most prominent video compression standards today is H.264, which is what we will be using. H.264 was developed as a response to the need for higher compression of moving pictures for several applications, such as

Internet streaming [11,12]. The main goal of this standard is to provide high quality video at much lower bit rates than previous standards, without increasing the complexity of the implementation. The aim is to make the standard flexible enough to be applied into a wide variety of applications, networks and systems. It has also been shown that H.264 clearly outperforms previous standards [13], partly due to the fact that many devices currently have built in hardware decoders for H.264. This improved compression performance entails a greater computational cost, enabling the H.264 to use significantly more processing power for the encoding and decoding. Pereira et al. present an analysis of the suitability of the H.264 standard for encoding IoT related video data for a low power personal area network in [14,15]. Worth noting, is that there is a successor to H.264 called High Efficiency Video Coding (HEVC) [16]. The HEVC codec does, however, lack hardware decoders built into the typical resource constrained IoT devices such as the Raspberry Pi.

We aim to use the H.264 standard for the coding of video data in our proposal because of the characteristics and simplicity of the H.264 byte stream format, and because of the hardware decoders on the Raspberry Pi devices. H.264 defines structures like the Network Abstraction Layer (NAL) units, facilitating access of the data within a stream. The first byte of each NAL unit is a header byte, which indicates which kind of data is present in the unit. The remaining bytes contain the payload data of the type indicated in the header. We will send these raw NAL units over an IoT communication system, directly from source to sink. The sink will then, on the fly, assemble the NAL units, decode the stream, and display the video. This approach should yield the lowest possible delay due to minimum buffering and stream parsing.

2.1.2. Internet of Things Communication Systems

Currently, there is a vast number of different systems used to connect IoT applications to sensors and actuators. Most are typical cloud-based systems with one or more centralized servers on the Internet, such as Nimbits, Azure IoT, Serviocity, Evrythng, Dweet, and Thingsquare [17]. These cloud-based systems are far from optimal when it comes to creating a future proof and ubiquitous IoT system [18], especially when it comes to large-scale communication, low delays, and avoiding central points of failure. Our approach will be a fully distributed and peer-to-peer approach, because the traditional cloud-based systems will have difficulties keeping the delay low, since all data need to be proxied through the cloud. Cloud systems also add a significant delay compared to true P2P communication, since P2P communication in its rawest form always takes place directly between source and sink. The IoT communication system we use will send the video stream directly from source to sink without any unnecessary proxying and without any intermediate nodes, creating a stream of data and achieving as low a delay as possible for the communication.

Most fully distributed IoT systems create an overlay using a DHT to enable logarithmic or better scaling when the participants increase in magnitude. There is some communication overhead related to the maintenance of the DHT itself, since it needs to maintain references between the participants of the DHT. In this paper we will focus on one of these DHT based systems, namely the SensibleThings platform [19], which is a fully distributed open source platform for enabling IoT applications supporting P2P communication with low overhead. There are several other IoT communication platform options. For example, closely related work is being done as a part of the TerraSwarm project [18]. Their solution is based on a data centric approach which can create unnecessary overhead since all new data is appended to long chains of distributed objects. The RELOAD architecture is another related work [20], which is also based on a fully distributed P2P system, but the solution uses the Session Initiation Protocol (SIP) which induces unnecessary overhead when applied to IoT scenarios and devices. There is also the Global Sensor Network (GSN) [21], but they have limited support for video streaming since originally, it aimed to connect data from wireless sensor networks to the Internet. Other relevant competing work preserves a cloud type system but moves it closer to the end users, so-called fog computing [22]. The response times are significantly lowered, but never become as low as for true P2P communication.

2.2. Our Method and Approach

Our approach is based on sending H.264 baseline coded NAL units as P2P packets over the SensibleThings Platform using Raspberry Pi 2 model B devices with attached camera modules. We selected this particular hardware to verify that our approach is viable for typical IoT devices. The SensibleThings platform was chosen since it is an openly available middleware platform for creating distributed IoT applications, capable of very low delay communication. An overview of our implementation can be seen in Figure 1; a Raspberry Pi 2 model B device with an attached camera module as the video source, the SensibleThings platform which will communicate the video data in a P2P manner, and finally a second Raspberry Pi 2 model B device, which will act as the video sink and render the video stream on a display connected via High-Definition Multimedia Interface (HDMI).

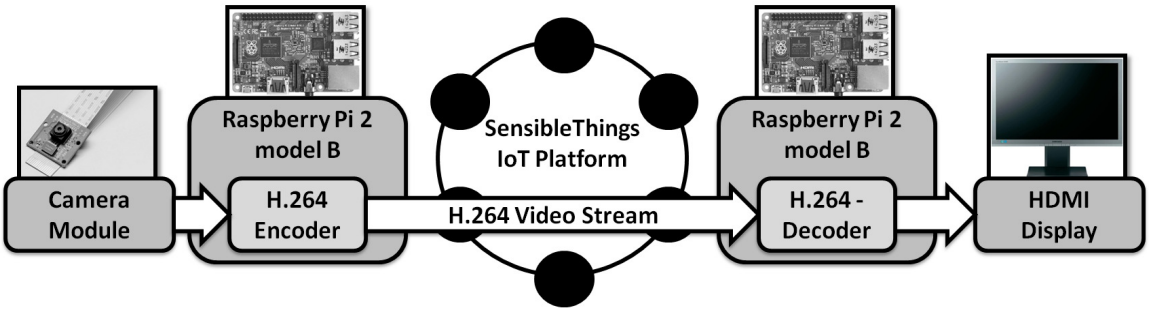

Figure 1. An overview of the implementation.

2.2.1. The Raspberry Pi Devices

The Raspberry Pi 2 model B [23] is a small computer with video and media coding capabilities. It was mainly developed for the promotion of computer science teaching in schools. It is, however, also used by a large community of different people and companies/organizations. The Raspberry Pi 2 model B has a Camera Serial Interface (CSI-2) connector to attach a camera module directly to the Broadcom VideoCore 4 Graphics Processing Unit (GPU) using the CSI-2 protocol. The camera is a high definition camera, capable of producing H.264 video using the hardware encoder built into the GPU. The camera can be controlled by an application called raspivid, which takes full advantage of the hardware encoding capabilities.

2.2.2. The SensibleThings Platform

The SensibleThings platform [19] is an open source communication platform enabling IoT based applications. The platform offers an open source framework for connecting sensors and actuators, in order to enable scalable real-time communication between applications. The main characteristics of the platform is that it scales logarithmically with communication load in the end-points; it has no central points of failure; it is capable of signaling in real-time between end points in a peer-to-peer manner, it has the ability to run on mobile devices with limited resources; and it is able to reliably handle transient nodes joining and leaving with high churn rates.

In order to find the sensors and actuators, the SensibleThings platform uses a DHT, where it associates the IP address of the sensors or actuators with a Universal Context Identifier (UCI). The UCI is akin to a combination between a Universal Resource Identifier (URI) and an e-mail address. For example, a temperature sensor belonging to a specific person could have the UCI: stefan.forsstrom@miun.se/temperature. The platform can also encrypt the P2P communication using the Secure Sockets Layer (SSL) protocol to prevent eavesdropping. If SSL is enabled there will, however, be additional overhead to the communication both in terms of delay and computational complexity, since SSL uses a six way handshake and both symmetric and asymmetric key calculations. A camera can be seen as a sensor, and the video stream can be considered a large set of continuous

sensor values to be sent over the platform. In this way we have taken advantage of all the peculiarities of the platform and can transfer the video stream from sources and sinks globally connected on the Internet in a very low delay P2P manner.

2.2.3. The Video Stream

The video stream is encoded using the H.264 encoder present in the Raspberry Pi 2 model B hardware. The H.264 compression offers enough performance to address any bandwidth concerns. Other video stream issues, such as packet loss and delay variation, are addressed by the IoT communication platform. For example, the SensibleThings platform ensures ordered reception of packets because of its reliable transmission protocol.

The operation starts with the session initialization, in which the two devices makes the initial connection. This includes the resolving of the camera name in the IoT platform and sending the initial get request for the video source to start the streaming. The application on the source side obtains the byte stream from the video encoder which in our implementation is solved by the raspivid command. Next, the source side will send an encoded NAL unit to the sink over the IoT communication platform. On the sink side, the application will retrieve the NAL units, one by one, and push them to the hardware decoder, which will decode and render the video stream on the display.

3. Results

There are many different methods for measuring the delay of streaming video. The easiest way is to observe both the video capturing and video displaying and compare them manually by inspecting triggers in the video. However, there are also more automated tools such as VideoLat [24], vDelay [25], and AvCloak [26]. To measure the low delay performance of our system, we have set up a simple manual measurement testbed. The measurements were made using a simulated digital clock (with an accuracy of one millisecond) on a laptop screen (HP EliteBook 8460p, Hewlett-Packard, Palo Alto, CA, USA) and a Raspberry Pi 2 model B with an attached camera (Raspberry Pi NoIR Rev. 1.3, Raspberry Pi Foundation, Cambridge, UK) facing the clock. It recorded and encoded the digital clock at a resolution of 1280 $\times$ 720 at 25 FPS, with the default bitrate setting for the raspivid application, creating a video stream with a bitrate of 6 mbit per second. The exact raspivid command used in all our measurements were: raspivid -n -vf -hf -ih -w 1280 -h 720 -fps 25 -t 0 -o -.

The recorded clock was then displayed on the second display (Samsung SyncMaster SA450, Samsung, Seoul, South Korea) to be compared with the live clock. This comparison was possible as the two displays were recorded simultaneously with a 300 FPS camera, and saved for later analysis. The complete system delay video could be calculated by comparing the clock difference, which was done by investigating the recorded still frames of the two screens. A figure displaying the resulting view of the two displays can be seen in Figure 2.

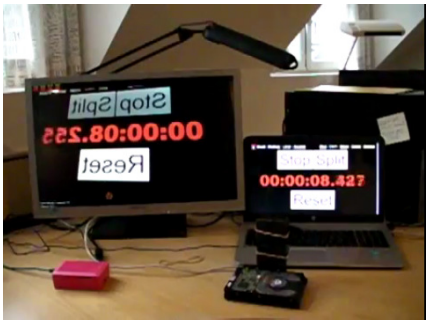

Figure 2. The recorded view of the two displays.

These measurements could then easily be repeated and the scenario altered by changing the network and device configurations. This was one of the reasons why we chose a manual measurement

testbed, because it would allow us to control all the steps in our evaluation and easily change the measurement set up. Another reason why we chose to manually record the digital clocks and compare them, rather than using a network packet analyzer or any other automatic methods to measure the video delay, is that we wanted to measure the complete delay from recording to display. A network packet analyzer can, for example, only measure on network level; it would not have included measurements of the encoding, decoding, and display delays.

3.1. Measurement Configurations

Five different measurement configurations were used to investigate different scenarios. This in order to isolate where the different delays came from and determine to what extent certain parts contributed to the delay as a whole. We chose to only use wired connections in all the scenarios, since the Raspberry Pi 2 model B does not have any wireless interfaces unless additional hardware is attached. The different configurations were: Capturing only, without network, with local network, SensibleThings with public IP, and finally SensibleThings with Network Address Translation (NAT) IP.

3.1.1. Capturing Only

In the first measurement setup we only measured the capturing time. This was done by using the "Preview" option on the Raspberry Pi 2 model B device, with the camera recording the digital clock. The video was shown directly on the connected screen in order to isolate the capture delay. See Figure 3 for an overview of how this measurement was set up. The measurements performed in this configuration showed that the capturing delay of the Raspberry Pi 2 model B device was on average 86.6 ms with a standard deviation of 0.713 ms.

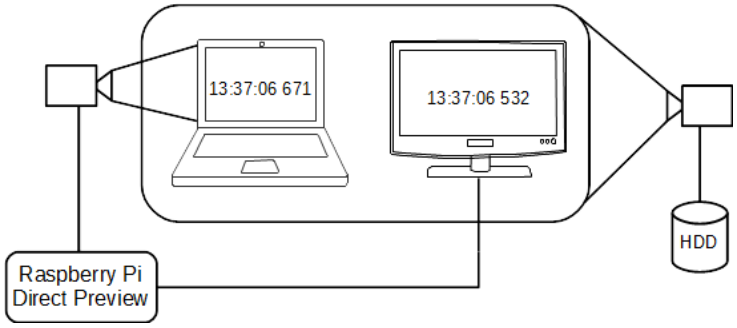

Figure 3. Measurement setup for capturing only.

3.1.2. Without Network

In the second setup we created a pipe to redirect the encoded video to a decoder which decoded the video on the same device. Both this measurement and the previous measurement were performed locally on a single device. This could therefore isolate the encoding and decoding delay. See Figure 4 for an overview of how this measurement was set up. The measurements performed in this configuration showed that the encoding and decoding delay of the Raspberry Pi 2 model B device was on average 163 ms with a standard deviation of 18.4 ms.

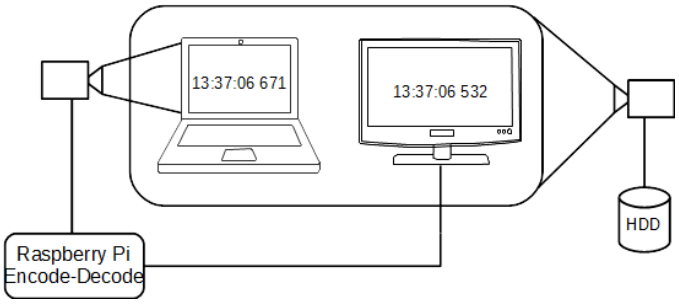

Figure 4. Measurement setup without network.

3.1.3. Only Local Network

The third measurement was similar to the second setup, but it used two different Raspberry Pi 2 model B devices and sent the video over a local gigabit network created by the local wired network of a Linksys WRT54GL v1.1 router (Linksys, Irvine, CA, USA). The encoded video was streamed directly using Transmission Control Protocol (TCP) from the Raspberry with the camera to the Raspberry with the screen. The receiving Raspberry then fed the received data directly to the decoder and displayed it on the screen. See Figure 5 for an overview of how this measurement was set up. The measurements performed in this configuration showed that the total delay of encoding and decoding on two Raspberry Pi 2 model B devices with a network between them was on average 163 ms with a standard deviation of 18.5 ms. This was very close to the previous measurement without the network, which indicates that the network communication itself does not add any significant overhead if it is on a local gigabit speed network.

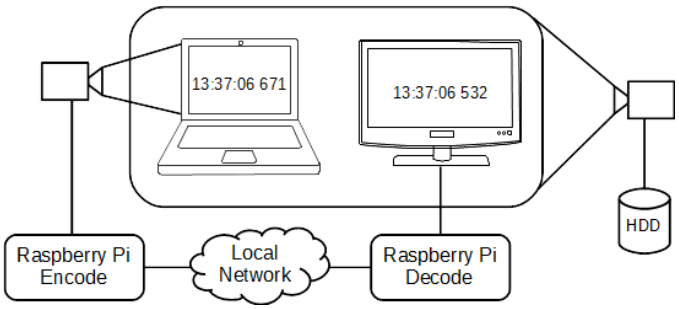

Figure 5. Measurement setup with local network.

3.1.4. SensibleThings and Public IP

The fourth measurement was done over the public Internet with the SensibleThings platform as communication method. In this measurement each of the Raspberry Pi 2 model B devices had a unique public IP address, as if they were directly connected to the Internet, without any home routers or firewalls. Both devices were connected with a 100/100 mbit connection to the same Internet service provider, namely the Swedish University Computer Network (SUNET). See Figure 6 for an overview of how this measurement was set up. This scenario is, however, not particularly realistic since public IP addresses are now quite uncommon and rarely issued to these types of end user devices. The measurements performed in this configuration showed that the delay of the Raspberry Pi 2 model B device when on the public Internet with the SensibleThings platform was on average 172 ms with a standard deviation of 11.0 ms. Indicating that the SensibleThings platform and the noise on the public Internet added roughly 9 ms on average to the delay.

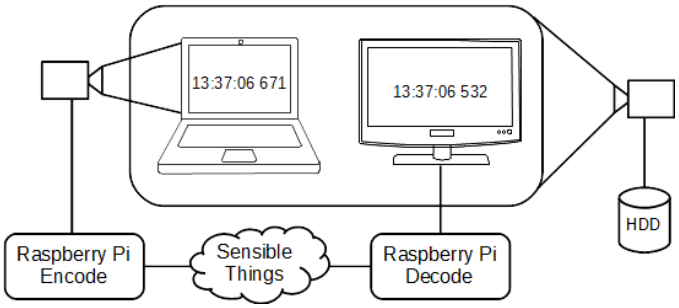

Figure 6. Measurement setup with SensibleThings with public Internet Protocol (IP).

3.1.5. SensibleThings and NAT IP

The final measurement scenario was in as realistic a setup as possible. In this scenario, both Raspberry Pi 2 model B devices were behind an NAT router and had private IP addresses. The SensibleThings platform had to apply different NAT penetration techniques to enable the P2P communication. This is a quite likely scenario, since most IoT devices will be behind home routers or behind a mobile carrier's router or firewall. The NAT networks were created using the Linksys WRT54GL v1.1 router connected to the public Internet via the 100/100 mbit SUNET connection. See Figure 7 for an overview of how this measurement was set up. The measurements performed in this configuration showed that the delay of the video streaming in this more realistic scenario was on average 181 ms with a standard deviation of 19.2 ms. Indicating that the added layer of NAT and complexity of the communication increased the delay further by roughly 9 ms.

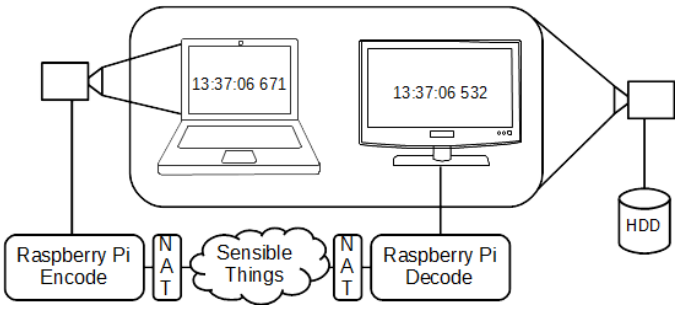

Figure 7. Measurement setup with SensibleThings with Network Address Translation (NAT) IP.

3.2. Measurement Summary

Our results are summarized in Table 1, showing the average video latency and standard deviation for each of the measurement setups. The raw data of all our measurements can be found online at: http://dx.doi.org/10.5281/zenodo.60681.

Table 1. Summary of all measurements.

Configuration	Average Delay	Standard Deviation
Capture only	86.6 ms	0.713 ms
Without network	163 ms	18.4 ms
Only local network	163 ms	18.5 ms
SensibleThings with public Internet Protocol (IP) address	172 ms	11.0 ms
SensibleThings with network address translation (NAT) IP	181 ms	19.2 ms

The summary shows that a fully distributed IoT platform such as the SensibleThings platform only accounts for 10% of the total delay in the worst case scenario, such as the more realistic scenario with

devices behind NAT, where the average delay was increased by 18 milliseconds. Another notable result is that a significant part of the delay is related to the video capture and encoding/decoding. We do not directly address the issue of scalability in this article. However, we can see that the system will scale well if both the communicating devices use public IP addresses, because the video is transferred in a peer-to-peer fashion. If both devices are behind NAT then the video must be proxied through relays, thus increasing the load of the proxy devices and the communication links.

4. Discussion

This article focused on the problem of communicating high definition live multimedia for IoT applications in scenarios with low delay under the constraints of typical IoT devices and hardware. That this is possible was shown by sending H.264 NAL units over a P2P-based IoT communication system on a typical IoT device. This article has also shown that our approach satisfies the three stated requirements. It has a low source to sink delay, which was requirement 1. We measured a 181 ms delay from source to sink, if both the source and sink are behind NAT networks. The transferred video was of a high definition quality of 1280×720 at 25 FPS, which was requirement 2. Finally it satisfies requirement 3, because it was shown to work on a Raspberry Pi 2 model B device, which can be considered a typical IoT devices with resource constrained hardware. In conclusion, when using a fully distributed IoT system 90% of the total delay is due to the encoding and to the decoding of the video.

Future Work

Real world deployment in the scenarios mentioned is our main future work, for example, to investigate the scalability aspects of our approach. In particular, there is a need to investigate the support and scaling for 50 billion devices which is the expected scale of the IoT. We also plan to make a survey of all the different IoT communication platforms and the interoperability between them. This includes measurements of the network capacities of the different IoT platforms and network architectures, especially to perform a quantitative comparison with streaming on typical cloud-based IoT systems and wireless networks. We would also like to evaluate other types of IoT devices, such as other types of single board computers, e.g., Raspberry Pi Zero and Raspberry Pi 3 and different smartphones. The impact of the video quality and contents of the streamed video was not considered in this work, hence the impact of realistic surveillance scenarios under different bitrate conditions and the implications on video quality is a relevant topic for future work. With new IoT devices released every year it would also make sense to investigate devices capable of HEVC (H.265). Security aspects are paramount to the proliferation of these types of IoT services and should therefore also be studied in more detail, especially if the IoT will be a reality in industrial and more critical scenarios. Finally, an industrial context also imposes other constraints, such as harsh physical and radio environment, that challenge the device shielding and wireless communication.

Acknowledgments: This research was supported by grant 20150363, 20140321, and 20140319 of the Swedish Knowledge Foundation.

Author Contributions: The article was written in collaboration between all authors. However, Ulf Jennehag conceived and designed the initial idea of the project, as well as lead the work. Stefan Forsstrom and Federico V. Fiordigigli realized the design, produced the implementation, and performed the measurements.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Atzori, L.; Iera, A.; Morabito, G. The internet of things: A survey. *Comput. Netw.* **2010**, *54*, 2787–2805.
2. Ericsson. *More than 50 Billion Connected Devices*; Technical Report; Ericsson: Stockholm, Sweden, 2013.
3. Cisco Systems Inc. *The Zettabyte Era: Trends and Analysis*; White Paper, Cisco Visual Networking; Cisco Systems Inc.: San Jose, CA, USA, 2014.

4. Alvi, S.A.; Afzal, B.; Shah, G.A.; Atzori, L.; Mahmood, W. Internet of multimedia things: Vision and challenges. *Ad Hoc Netw.* **2015**, *33*, 87–111.
5. Jiang, W.; Meng, L. Design of Real Time Multimedia Platform and Protocol to the Internet of Things. In Proceedings of the 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Liverpool, UK, 25–27 June 2012; pp. 1805–1810.
6. Martinez-Julia, P.; Torroglosa Garcia, E.; Ortiz Murillo, J.; Skarmeta, A.F. Evaluating Video Streaming in Network Architectures for the Internet of Things. In Proceedings of the 2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), Taichung, Taiwan, 3–5 July 2013; pp. 411–415.
7. Nguyen, H.Q.; Loan, T.T.K.; Mao, B.D.; Huh, E.N. Low cost real-time system monitoring using Raspberry Pi. In Proceedings of the 2015 Seventh International Conference on Ubiquitous and Future Networks (ICUFN), Sapporo, Japan, 7–10 July 2015; pp. 857–859.
8. Vamsikrishna, P.; Hussain, S.R.; Ramu, N.; Rao, P.M.; Rohan, G.; Teja, B.D.S. Advanced Raspberry Pi Surveillance (ARS) system. In Proceedings of the 2015 Global Conference on Communication Technologies (GCCT), Thuckalay, India, 23–24 April 2015; pp. 860–862.
9. Ansari, A.N.; Sedky, M.; Sharma, N.; Tyagi, A. An Internet of things approach for motion detection using Raspberry Pi. In Proceedings of the 2014 International Conference on Intelligent Computing and Internet of Things (ICIT), Harbin, China, 17–18 January 2015; pp. 131–134.
10. Biedermann, D.H.; Dietrich, F.; Handel, O.; Kielar, P.M.; Seitz, M. Using Raspberry Pi for scientific video observation of pedestrians during a music festival. 2015, arXiv preprint arXiv:1511.00217.
11. Wiegand, T. *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification*; ITU-T rec. H. 264 | ISO/IEC 14496-10 AVC; ISO/IEC: Pattaya, Thailand, 2003.
12. Wiegand, T.; Sullivan, G.J.; Bjøntegaard, G.; Luthra, A. Overview of the H.264/AVC video coding standard. *IEEE Trans. Circ. Syst. Video Technol.* **2003**, *13*, 560–576.
13. Kamaci, N.; Altunbasak, Y. Performance comparison of the emerging H.264 video coding standard with the existing standards. In Proceedings of the IEEE 2003 International Conference on Multimedia and Expo, Baltimore, MD, USA, 6–9 July 2003; Volume 1, pp. 345–348.
14. Pereira, R.; Pereira, E. Video Streaming: H.264 and the Internet of Things. In Proceedings of the 2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops (WAINA), Gwangju, Korea, 24–27 March 2015; pp. 711–714.
15. Pereira, R.; Pereira, E. Video Streaming Considerations for Internet of Things. In Proceedings of the IEEE 2014 International Conference on Future Internet of Things and Cloud (FiCloud), Barcelona, Spain, 27–29 August 2014; pp. 48–52.
16. Sullivan, G.J.; Ohm, J.R.; Han, W.J.; Wiegand, T. Overview of the high efficiency video coding (HEVC) standard. *IEEE Trans. Circ. Syst. Video Technol.* **2012**, *22*, 1649–1668.
17. Alamri, A.; Ansari, W.S.; Hassan, M.M.; Hossain, M.S.; Alelaiwi, A.; Hossain, M.A. A survey on sensor-cloud: Architecture, applications, and approaches. *Int. J. Distrib. Sens. Netw.* **2013**, *2013*, doi:10.1155/2013/917923.
18. Zhang, B.; Mor, N.; Kolb, J.; Chan, D.S.; Goyal, N.; Lutz, K.; Allman, E.; Wawrzynek, J.; Lee, E.; Kubiatowicz, J. The Cloud is Not Enough: Saving IoT from the Cloud. In Proceedings of the 7th USENIX Workshop on Hot Topics in Cloud Computing, Santa Clara, CA, USA, 6–7 July 2015.
19. Forsström, S.; Kardeby, V.; Österberg, P.; Jennehag, U. Challenges when Realizing a Fully Distributed Internet-of-Things-How we Created the SensibleThings Platform. In Proceedings of the 9th International Conference on Digital Telecommunications ICDT, Nice, France, 23–27 February 2014; pp. 13–18.
20. Jennings, C.; Baset, S.; Schulzrinne, H.; Lowekamp, B.; Rescorla, E. *Resource Location and Discovery (Reload) Base Protocol*; IETF: Fremont, CA, USA, 2014.
21. Aberer, K.; Hauswirth, M.; Salehi, A. *The Global Sensor Networks Middleware for Efficient and Flexible Deployment and Interconnection of Sensor Networks*; Technical Report LSIRREPORT-2006-006; Ecole Polytechnique Federale de Lausanne: Lausanne, Switzerland, 2006.
22. Bonomi, F.; Milito, R.; Natarajan, P.; Zhu, J. Fog computing: A platform for internet of things and analytics. In *Big Data and Internet of Things: A Roadmap for Smart Environments*; Springer: Berlin, Germany, 2014; pp. 169–186.
23. Upton, E.; Halfacree, G. *Raspberry Pi User Guide*; John Wiley & Sons: New York, NY, USA, 2014.

24. Jansen, J. VideoLat: An Extensible Tool for Multimedia Delay Measurements. In Proceedings of the 22nd ACM International Conference on Multimedia, Orlando, FL, USA, 3–7 November 2014; pp. 683–686.
25. Boyaci, O.; Forte, A.; Baset, S.A.; Schulzrinne, H. vDelay: A tool to measure capture-to-display latency and frame rate. In Proceedings of the 11th IEEE International Symposium on Multimedia, San Diego, CA, USA, 14–16 December 2009; pp. 194–200.
26. Kryczka, A.; Arefin, A.; Nahrstedt, K. AvCloak: A tool for black box latency measurements in video conferencing applications. In Proceedings of the 2013 IEEE International Symposium on Multimedia (ISM), Anaheim, CA, USA, 9–11 December 2013; pp. 271–278.