Towards More Adaptive Voice Applications

Jörg Ott

Aalto University
School of Science and Technology
Department of Communications and Networking
jo@comnet.tkk.fi

Abstract. With the Internet designed to provide best-effort packet transmission, applications are expected to adapt dynamically to the operating conditions observed in the network. For this purpose, congestion control mechanisms have been devised for various transport and (partly) application protocols, and application programs may present, e.g., data rate information to the user. While these mechanisms work well for elastic applications (such as file transfer), the perceived performance of real-time applications may degrade quickly if a minimum required quality of service cannot be achieved. We argue that the current interpretation of adaptation specifically of real-time applications is too narrow and present a framework for expanding the scope of end-to-end adaptation, using the case study of voice communications. Our approach is general in nature, but should especially support communication in mobile environments.

1 Introduction

The Internet Protocol inherently offers a best-effort datagram delivery service, allowing for arbitrary delay, reordering, loss, and duplication of packets. While reordering and duplication occur but are not (yet) commonplace, delays and losses are inherent properties used in the operation of many Internet protocols, as they serve as a measure for congestion. Transport and application protocols (should) monitor these values and are supposed to dynamically adapt their behavior to the changing network conditions.

While transport protocols may be designed to adapt as needed (see section 2), two related assumptions are implied for the applications involved: A1) They are capable of adapting across a wide range of transmission characteristics. In practice, however, the only truly adaptive applications appear to be file transfer, be it as a simple client-server system or a more sophisticated peer-to-peer sharing application, and other ones operating in the background without being time critical. Instead, most applications have a limited operational range of network characteristics acceptable to the user, leading to: A2) The best effort service achieved will be sufficient for the application needs. If it is not, users will notice and react and typically stop using the applications. This applies to (semi-)interactive TCP-based applications (such as web access, HTTP streaming, and SSH) and even more so to UDP-based real-time applications such as VoIP.

We use the term *elastic applications* for those that are capable of adapting as described above (at least within very broad limits), and *inelastic applications* to denote those that are limited to a narrower range of operating conditions: due to their very nature (e.g., real-time monitoring) or because of (present) user expectations (e.g., conversational voice).

In this paper, we broaden the scope of application adaptivity in two ways using voice communication as a case study: a) We capture a broader range across which to adapt, specifically extending to high delays and temporary disconnections so that networking conditions are less critical for a voice conversation. b) To support this broader range, we include more of the application semantics into the considerations of the adaptation process: In our specific case, we embrace asynchronous voice messaging and semi-synchronous (two-way alternate) walkie-talkie-style communications and make these become an integral part of a voice conversation as we will discuss below.

Our goal is to conceptually capture those cases in which the above assumption A2) no longer holds and to devise exemplary mechanisms to fulfill assumption A1) nevertheless. After a short review of related work on adaptive communication (focusing on congestion control) in section 2, we describe mobile communication environments as one case in which assumptions A1) easily fails in section 3. We present the case study for voice communications in section 4, using two examples of our prior work addressing how to maintain A2), from which we then develop a more general adaptation mechanism. We conclude with a brief discussion in section 5.

2 Background and Related Work

At the transport layer, the notion of congestion control has been introduced to TCP [Jac88] (with countless variants developed since) to share network resources fairly (at the flow level), a concept incorporated in recent protocols such as SCTP [Ste07] and DCCP [KHF06a,KHF06b]. Transport protocols also perform timeout adaptation (e.g., TCP's RTO) to cope with varying delays across different networks, which may span more than six orders of magnitude between a local Ethernet link and a loaded GPRS network. Path MTU discovery [MD90] is used to determine an appropriate packet size (e.g., TCP's segment size) in order to avoid fragmentation, although today often 1460 bytes are assumed to be safe.

All these mechanisms are hidden inside the transport layer and thus invisible to the applications (figure 1 (left)). The underlying assumption is that applications using such transport protocols are sufficiently elastic, i.e., can deal well with changing transmission rates and delays. This abstraction was criticized in the past (e.g., [MBL$^+$04]), as TCP would retransmit potentially outdated information and an application is not able to determine the detailed status of the send buffer. The channel concept of SCTP and its unreliable mode, developments such as *Structured Streams* [For07], and more direct resource control [MBL$^+$04] would allow for some more flexibility. Irrespective of these limitations, several applications use TCP to carry real-time data (e.g., skype uses TCP as a fallback) to simplify/enable NAT traversal, with some adaptation realized on top [GKLW02,BBRS08].

Inelastic applications, in contrast, have often avoided TCP and realized the necessary protocol mechanisms on top of UDP, leaving full control to the application as shown in figure 1 (right). This applies to performing semantic fragmentation of application data units and their mapping onto packets [CT90] as is common for real-time

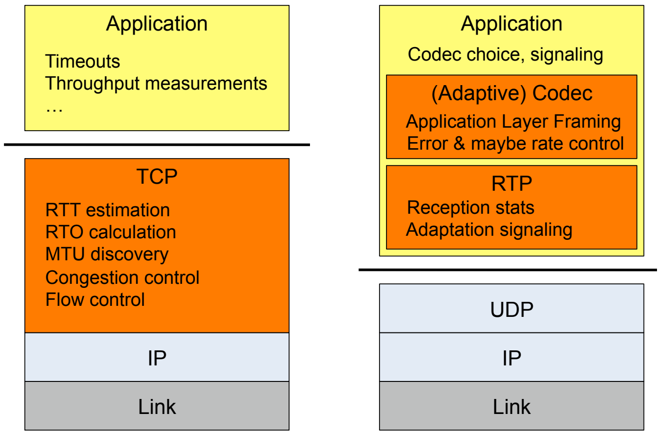

Fig. 1. Extreme cases for adaptivity in today's protocol stacks

media transmissions using RTP [SCFJ03,HP99]. Real-time applications are also supposed to adapt to network congestion and packet losses by means of rate adaptation and error repair and concealment techniques. Rate adaptation can be performed, e.g., by switching codecs [BVG96], changing codec parameters as for multi-rate codecs such as AMR [SWB01], and by adjusting packet sizes. Error control may involve interleaving [PHH98], dynamically applying FEC [PKH$^+$97,Li07,BFPT99] or retransmissions [RLM$^+$06,KH04,ÁHI08]. Several of the above mechanisms may also be combined (e.g., [MPLJ03]).

The necessary feedback loops for observing network conditions to allow a sender to dynamically adapt its transmission behavior may be based upon RTCP reporting [SCFJ03,OWS$^+$06,Gar07]. Such feedback loops may also use information of transport protocols such as DCCP [Per07,BENB07] (or SCTP), in which case the solutions are in-between the two extremes shown in figure 1.

Overall, the key consideration for adaptivity in interactive real-time communications remains the perceivably acceptable delay when trading off delay, loss, and data rate in rate control, interleaving, FEC, and retransmission schemes. According to ITU-T G.114 [Int], the one-way delay for interactive voice communication should not exceed 150 ms, 150–400 ms are potentially tolerable, and delays above 400 ms are not acceptable. This limits the range of network conditions across which these applications can operate *if the notion of interactive voice shall be preserved*.

3 The (Mobile) Internet Today: When Best Effort Is Not Enough

With wireless and mobile Internet becoming increasingly dominant, the connectivity characteristics for mobile users deserve explicit consideration in protocol design [Ott06]. It is well known that wireless and mobile connectivity characteristics may differ significantly from what is observed in the fixed Internet. Specifically, wireless links are susceptible to attenuation, interference, etc. as well as varying load due to shared channels and may yield highly variable delays (jitter) due to link layer retransmissions and queuing. Worse, the physical layer phenomena and also coverage gaps may lead to temporary loss of connectivity: from fractions of a second to minutes or hours.

In effect, loss, delay, and throughput of wireless links may vary more heavily and more abruptly than in fixed networks, making adaptation more demanding. While congestion in fixed networks often builds up gradually and thus makes rate, loss, and delay estimation somewhat predictable (unless routing changes), sudden interference, handovers to networks with (much) lower performance, or coverage gaps may impact (and possibly stall) communication instantaneously; similarly, conditions may improve suddenly and capacity becoming available again may not be fully exploited.[1] To cope with short disconnections (specifically during handovers), network operators often perform significant queuing for their mobile data networks; however, this leads to delay being accumulated, negatively impacting TCP and interactive application protocols. On the other hand, if no or little queuing takes place, more losses occur. Longer disconnections will lead to packet losses in either case. Overall, the outcome from an application perspective is delay [Ott08].

Interactive voice applications suffer from high delays and jitter as discussed above. Jitter requires playout buffering and thus may incur additional delays (if adapted too conservatively); when delays exceed the predictions for playout buffering so that packets miss their playout deadline, these packets are discarded, increasing loss.

In contrast, VoIP can adapt quite well to the available bit rate. Since a range of codecs exists nowadays (some of them support variable data rates), audio communication can be adapted between some 4 kbit/s (Speex, AMR-NB) and 64 kbit/s (G.711), providing a flexible operating range to cope with congestion. If the short-term average data rate drops below the lower bound, transmitting speech is no longer possible in real-time and additional delay is introduced—or packet losses occur. However, delay and round-trip time (RTT) impact the reactivity of the rate adaptation mechanisms and late adaptation may increase queuing delays and/or cause losses.

Occasional packet losses (due to congestion or bit errors) are tolerable for voice applications and can be handled by receiver-side error concealment techniques. More substantial loss rates or loss bursts lead to unintelligible speech and therefore require applying error control mechanisms, all of which increase the overall delay. As it was found for *skype* users that delay has a less significant impact on perceived speech quality than losses [CHHL06], some flexibility for packet repair exists beyond the guidelines of G.114 [Int], so that loss bursts and short-term outages might be concealed.[2]

[1] From an endpoint's perspective it is close to irrelevant whether communication is inhibited by congestion or due to mobility (and it may be impossible to find out); hence, a protocol instance at an endpoint should not (need to) care.

[2] It is worthwhile noting that ITU-T recommendation G.114 is from 1993, a time before the widespread acceptance of mobile telephony (with quite different reliability characteristics) and VoIP. User expectations are probably changing with the use of different communication infrastructures and users may be willing to trade off, e.g., the earlier notion of quality for additional value (mobility, reachability, presence and messaging integration) or lower cost.

Nevertheless, as losses, delays or the frequency of disconnections grow, interactive voice becomes essentially unusable. At some point, rate and error control mechanisms may no longer be capable of adapting to the environment and the user has to choose between less interactivity and reduced intelligibility.

This example shows that wireless and mobile communication may yield a best effort service that is insufficient for specific applications if the network characteristics are (temporarily) outside the operational range supported by the respective application (in our case conversational voice). In the above example, the voice communication application is limited in its capability to adapt to a broader range of operational conditions because of the fairly strict interactivity requirements: the specific application semantics define what *interactive voice* means to meet the user expectations.[3] Rather than trying to make all networks match the application needs (e.g., using elaborate QoS mechanisms), we suggest to revisit the application semantics and investigate whether those can be broadened to allow expanding the application's operational range.[4]

For voice communication, this could mean reconsidering the notion of *interactivity*: if we are willing to tolerate higher one-way delays well beyond 150–400 ms, we may be able to employ better error control mechanisms and improve intelligibility of speech. And we may be able to deal with (short) connectivity disruptions by temporarily buffering speech and relaying it when connectivity becomes available again. Both would shift voice communications towards less synchronous interactions if the environmental conditions so require. In the following section, we will discuss two distinct extensions to conversational voice applications that explore the aforementioned ideas. We then develop them further towards a concept of a more adaptive voice application that has a broader notion of interactive voice.

4 Case Study: Adaptive Voice Communication

Users of mobile phones are well aware of varying network conditions and the resulting effects ranging from short-term outages to call disconnections. And users of VoIP over wireless networks (such as WLANs) often experience (short) periods of unintelligible speech due to losses. In either case, the voice applications are not capable of adapting to the network conditions and error handling is up to the user: from "manual synchronization" by repeating sentences to re-dialing a lost call [OX07]. We have investigated two different classes of mobile voice communication applications to deal with insufficient connectivity:

[3] For other applications, this holds even if the underlying transport (and session) protocols are capable of tolerating a broader operational range (see, e.g., the countless extensions on TCP performance enhancements and disconnection tolerance as discussed in [OSC$^+$09]) and if the user would be willing to tolerate a lower degree of interactivity, simply because application operations may time out when disconnections last sufficiently long.

[4] Recall that, due to the very nature of wireless communications, QoS cannot always be guaranteed, so that the mechanisms discussed here should also be suitable for well-managed wireless networks.

1. **Disconnection-tolerant SIP**

   We have investigated switching between plain SIP voice calls and voice messaging for IP networks, integrating automatic redial functionality [OX07]. In this approach, two endpoints constantly monitor the networking conditions by observing RTP and RTCP reports from their respective peer to detect gaps in connectivity. These gaps are classified into short (up to several seconds of lost speech, but the call continues), medium (up to one minute, call disconnected), and large (more than one minute, call disconnected and could not be re-established). In addition, each endpoint records the last few seconds of transmitted speech in a local ring buffer for later auto-recovery.

   Short outages are recovered by automatically repeating the last (likely incompletely received) talk spurt(s) to the peer (which performs duplicate filtering to avoid too much replay) so that the re-synchronization ("What did you say?") is not entirely up to the users. Medium gaps are repaired by automated re-dialing and answering paired with playback so that the parties can continue talking. Long disconnections are addressed by redirecting the last bits of the conversation to the peer's voice mail so that at least the last statements can be completed and stored for later retrieval.

   These mechanisms address those cases in which the interactive conversation is already disturbed by the environment and provide a means to simplify recovery.

2. **DT-Talkie: Asynchronous Voice Messaging**

   We have developed an asynchronous voice messaging application running on top of delay-tolerant networks (DTNs) [Fal03] that allows peer-to-peer voice conversations for two or more parties, somewhat similar to Push-to-Talk services (PTT), but without reliance on network or server infrastructure and without the need for a real-time path [ITK$^+$09,Isl09]. Its use of DTN concepts makes *DT-Talkie* applicable to ad-hoc network environments as well as disconnection-prone mobile connectivity.

   The *DT-Talkie* captures and stores voice statements locally and groups these statements into voice messages. The messages are sent asynchronously to the respective peer, directly or indirectly via intermediate nodes, with each message being transmitted reliably hop-by-hop using the underlying DTN protocol stack. After (complete) reception, the receiver renders the message from local memory; timing is maintained within but not across statements. Capturing to and rendering from local memory ensure smooth recording and playback, the use of a hop-by-hop-reliable transport ensures that no message parts get lost. Together, this decouples the fidelity of voice communication (perfect intelligibility) from the underlying network conditions, at the expense of increasing delay.

   This application goes well beyond disconnection-tolerant SIP in assuming non-connectivity in the first place. Well-connected nodes experience roughly similar quality as Push-to-talk-over-Cellular (PoC) services, whereas poorly connected ones are able to communicate better than before.

We can generalize these ideas further and synthesize both approaches if we assume a more flexible communication substrate allowing for the exchange of small to arbitrarily large packets (or: messages)[5], thus creating a continuum of voice-based interactions as shown in figure 2. At the top of the figure, we indicate today's disjoint voice applications from real-time interactive voice on one end of the spectrum to voice mail on the other. These disjoint applications represent different ways of interactions between users with different degrees of interactivity and delay tolerance. They could be integrated as different *modes of operation* into a single encompassing voice application as described below.

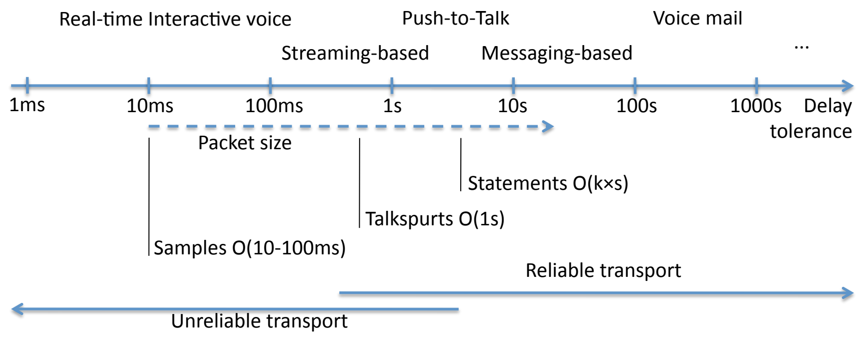

Fig. 2. Adaptivity of voice applications: Different application classes (top) exhibit different degrees of delay tolerance (shown on the axis)—which is related to their packet sizes (shown using the order notation O(...). The larger the packets are, the more important gets reliable delivery.)

As noted above, a key metric for communications is delay, since packet loss may be reduced when allowing for more delay, disconnections can be overcome by waiting sufficiently long, and the data rate of audio codecs appears sufficiently adaptive for most scenarios; if less instant capacity is available, reverting to non-real-time transmission will help, at the expense of increased delay. The mode of operation with the lowest delay are interactive real-time voice conversations, in which we assume minimal mouth-to-ear-delay (typically < 150ms) for acceptable interactivity. Longer delays are tolerable if we move towards less synchronous interactions, as known from one-way alternate communications via walkie-talkies or *Push-to-Talk*; yet some degree of interactivity is preserved. Depending on the network conditions, information exchange may be based upon real-time packet streaming (as in Push-to-talk over Cellular) or reliable exchange of messages (as in our *DT-Talkie*). Finally, voice messaging offers a rather asynchronous style of interaction, more comparable to email.

If we exploit all these different modes of operation and move smoothly between them (subject to the consent of the user), we can make VoIP applications more elastic and expand their operational range. This is conceptually depicted in figure 3. The

[5] This functionality may be realized at the network layer, as in some DTNs or in a future Internet [Ott08], or at the transport or application layer as an overlay on top of IP.

traditional adaptation for (in this case SIP-based) VoIP systems is shown at the top: The user has some high-level control to specify preferences (usually via quite flexible default settings to ensure interoperability) towards the VoIP application. It couples the media exchange and the signaling functions (call setup and teardown, etc.) and performs the real-time capturing and rendering functions. Based upon the user preferences, the codec and transport layers (their mechanisms are often also referred to as source and channel coding, respectively) perform codec-specific and generic error and rate control. Assuming, e.g., RTCP to monitor RTT, jitter, loss rate, and (implicitly) connectivity, the endpoints can adapt their bit rate by choosing different audio codecs and their packet rate (and header overhead) by varying packet sizes.

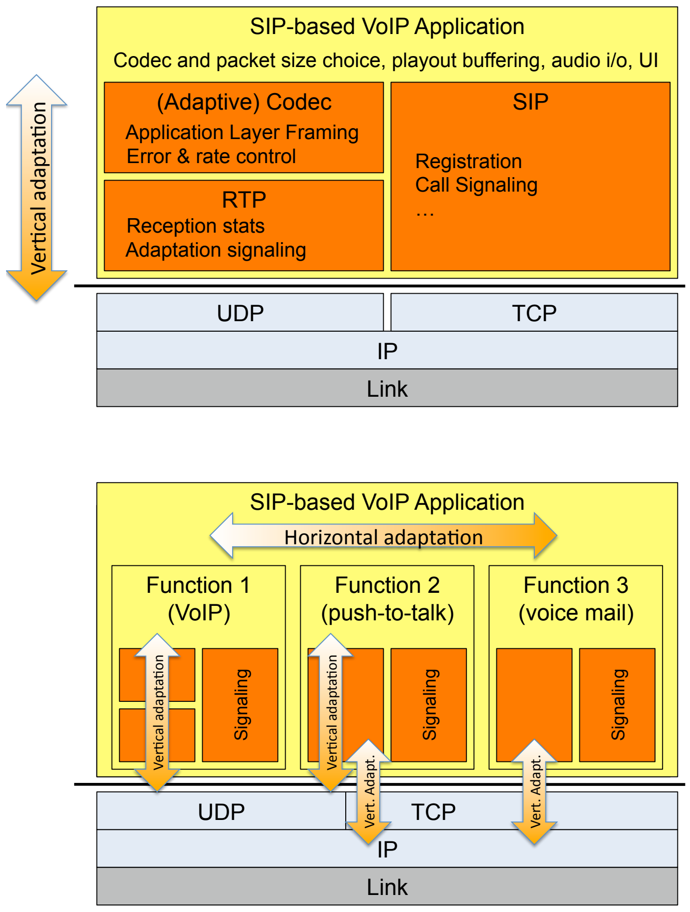

Fig. 3. Vertical only (top) and integrated horizontal (bottom) voice adaptivity

The entire adaptation is *vertical*, i.e., constrained to the media protocol stack, and after the initial call setup not further intertwined with call signaling. Similar vertical adaptation mechanisms exist for other types of voice interaction as shown in the figure at the bottom: however, no cross-function adaptation is foreseen in today's applications.[6]

The above vertical adaptation mechanisms can be leveraged when broadening the scope of interactive voice communications and smoothly extending it to cover push-to-talk- and voice-message-style communications as well. This is conceptually shown in figure 3 at the bottom, denoted as *horizontal* adaptation. Depending on the observed network conditions, an adaptive application may move between different modes of operations. Ideally, there are no fixed boundaries as in today's applications, but rather a smooth transition would be foreseen.

As shown in figure 2, starting out with a regular voice call, packets are kept small to achieve a high degree of interactivity, with the above vertical adaptation mechanisms applied. When network conditions worsen, voice application data units (ADUs) can be increased further. With increasing ADU sizes, however, delay increases and communication loses interactivity. While using regular sampling intervals (e.g., 10–100ms) for smaller packets, an application may decide to identify talk-spurts and send (groups of) talk-spurts in ADUs to keep such related information together: words of a sentence and a sequence of sentences of a statement. This ensures that related pieces of each talk-spurt are either delivered in their entirety and can be played back without interruption or are not delivered at all. This can be expanded further to gather complete statements of a user (as with walkie-talkies), e.g., by means of local processing using heuristics, leading further towards asynchronous communication. If connectivity to a peer is lost entirely (for some time), one or more local statements may be aggregated and turned into voice mail. Continuous monitoring of networking conditions should also indicate when the situation is improving, so that the application can move again towards synchronous operation (or resume communication after disconnection).

With increasing ADU size, the impact of a single lost ADU grows: error concealment will work less well for 100ms of missing speech than for only 20ms and if entire statements get lost, the peer may wonder why nobody is responding. Hence, larger ADUs suggest using more reliable transport mechanisms—which are "affordable" since the acceptable delay also increases, thus e.g. allowing for retransmissions.

5 Discussion and Conclusion

The above example presents a conceptual *technically-oriented* view on adaptive applications. We have outlined how broader adaptation could be achieved and sketch how media transmission, monitoring, and to some extent signaling could interact to realize this idea. But the details require further investigation and specific protocol and system designs will need rigorous analytical, simulation, and experimental validation. An interesting technical challenge will be multi-party conversations with the parties connected to each other under different, varying path conditions.

[6] Of course, a call setup request may be directed to a voice mailbox; but this happens at the call setup time, before the conversation starts, and is thus not related to adaptation.

At least equally important, however, is the *non-technical* dimension. While it may be feasible to design such an encompassing adaptive voice application, will the idea of a smooth transition across a very broad operational range be *accepted* by human users? This involves at least two aspects: *usability* and *user expectations*. Concerning the former, a suitable (intuitive, unobtrusive) user interface is required. It has to ensure smooth motion back and forth between more synchronous and more asynchronous styles of interaction, with right level of reactivity. It should offer embedded cues to the user about the present degree of interaction for a given conversation, and it should allow a user to easily control the range of adaptation acceptable for a given conversation. As for the latter, it appears important to steer user expectations: this kind of more adaptive application that is a phone at one instant and turns into a walkie-talkie at the next would probably not appeal someone expecting a phone that always works. Since we have seen in the aforementioned examples of mobile phones and *skype* that users are able to adapt expectations (including increased delay tolerance) and behavior, we are optimistic that also the idea of broader adaptation mechanisms could be embraced.

Finally, while we offered some intuition on how more comprehensive considerations could help adaptivity of voice communications, similar considerations may be applicable to other real-time and non-real-time applications: For example, media streaming applications already perform pre-buffering to deal with varying networking conditions, a concept that has been explored further to deal with temporary disconnections and could be extended towards broader adaptivity in general, possibly integrating Podcast-style downloads and real-time streaming. Also, web applications could toggle more smoothly between online and offline operation [OK06], provided that the application protocols are adapted accordingly [Ott06].

One interesting follow-up question is whether commonalities can be identified across different applications for common support in future transport protocols. Another one warranting further discussion is what the implications on the present (or a future) networking infrastructure are. We have argued that a future Internet should become inherently more delay-tolerant [Ott08] but, as we discussed above, there is a feature interaction between queuing/buffering ADUs inside the network and the end-to-end control loops of the applications. This may call for limited additional interaction between endpoints and network elements—e.g., providing hints for ADU processing—while maintaining the endpoints independent of the network elements.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement no 216714.

References

[BBRS08] Eli Brosh, Salman Abdul Baset, Dan Rubenstein, and Henning Schulzrinne. The Delay-Friendliness of TCP. In *Proceedings of ACM SIGMETRICS*, pages 49–60, 2008.

[BENB07] Vlad Balan, Lars Eggert, Saverio Niccolini, and Marcus Brunner. An Experimental Evaluation of Voice Quality over the Datagram Congestion Control Protocol. In *Proceedings of IEEE INFOCOM*, 2007.

[BFPT99] Jean-Chrysostome Bolot, Sacha Fosse-Parisis, and Don Towsley. Adaptive FEC-based error control for Internet telephony. In *Proceedings of IEEE INFOCOM*, volume 3, pages 1453–1460, 1999.

[BVG96] Jean-Chrysostome Bolot and Andrés Vega-García. Control Mechanisms for Packet Audio in the Internet. In *Proceedings of IEEE INFOCOM*, volume 1, pages 232–239, 1996.

[CHHL06] Kuan-Ta Chen, Chun-Ying Huang, Polly Huang, and Chin-Laung Lei. Quantifying skype user satisfaction. In *Proceedings of ACM SIGCOMM*, pages 399–410, 2006.

[CT90] David D. Clark and David L. Tennenhouse. Architectural Considerations for a new Generation of Protocols. In *Proceedings of ACM SIGCOMM*, pages 200–208, 1990.

[Fal03] Kevin Fall. A Delay-Tolerant Network Architecture for Challenged Internets. In *Proceedings of ACM SIGCOMM*, pages 27–34, 2003.

[For07] Bryan Ford. Structured Streams: a New Transport Abstraction. In *Proceedings of ACM SIGCOMM*, pages 361–372, 2007.

[Gar07] Ladan Garai. RTP with TCP Friendly Rate Control. Internet Draft draft-ietf-avt-tfrc-profile-09.txt, Work in progress, July 2007.

[GKLW02] Ashvin Goel, Charles Krasic, Kang Li, and Jonathan Walpole. Supporting Low Latency TCP-Based Media Streams. In *Proceedings of IEEE IWQoS*, pages 193–203, 2002.

[HP99] Mark Handley and Colin Perkins. *Guidelines for Writers of RTP Payload Format Specifications*, December 1999. RFC 2736.

[Int] International Telecommunication Union, Telecommunication Sector (ITU-T). Transmission Systems and Media, General Recommendation on the Transmission Quality for an Entire International Telephone Connection; One-Way Transmission Time. Recommendation G.114, March 1993.

[Isl09] Md. Tarikul Islam. Voice Communications in Mobile Delay-tolerant Networks. Master's thesis, Helsinki University of Technology (TKK), Finland, 2009.

[ITK$^+$09] Md. Tarikul Islam, Anssi Turkulainen, Teemu Kärkkäinen, Mikko Pitkänen, and Jörg Ott. Practical Voice Communications in Challenged Networks. In *Proceedings of the ExtremeCom workshop*, 2009.

[Jac88] Van Jacobson. Congestion Avoidance and Control. In *Proceedings of ACM SIGCOMM*, pages 314–329, 1988.

[KH04] Michael Kropfberger and Hermann Hellwagner. Evaluation of RTP Immediate Feedback and Retransmission Extensions. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, volume 3, pages 1751–1754, 2004.

[KHF06a] Eddy Kohler, Mark Handley, and Sally Floyd. Datagram Congestion Control Protocol (DCCP). RFC 4340, March 2006.

[KHF06b] Eddy Kohler, Mark Handley, and Sally Floyd. Designing DCCP: congestion control without reliability. In *Proceedings of ACM SIGCOMM*, pages 27–38, 2006.

[Li07] A. Li. RTP Payload Format for Generic Forward Error Correction. RFC 5109, December 2007.

[MBL$^+$04] Jeffrey Mogul, Lawrence Brakmo, David E. Lowell, Dinesh Subhraveti, and Justin Moore. Unveiling the Transport. *ACM SIGCOMM Computer Communication Review*, 34(1):99–105, January 2004.

[MD90] Jeffrey Mogul and Steve Deering. *Path MTU discovery*, November 1990. RFC 1191.

[MPLJ03] Johnny Matta, Christine Pépin, Khosrow Lashkari, and Ravi Jain. A Source and Channel Rate Adaptation Algorithm for AMR in VoIP Using the Emodel. In *Proceedings of ACM NOSSDAV*, pages 92–99, 2003.

[OK06] Jörg Ott and Dirk Kutscher. Bundling the Web: HTTP over DTN. In *Proceedings of WNEPT*, 2006.

[OSC+09] Jörg Ott, Nils Seifert, Caleb Carroll, Nigel Wallbridge, Olaf Bergmann, and Dirk Kutscher. The CHIANTI Architecture for Robust Mobile Internet Access. In *Proceedings of the 10th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), Industry Track*, 2009.

[Ott06] Jörg Ott. Application Protocol Design Considerations for a Mobile Internet. In *Proceedings of the 1st ACM MobiArch Workshop*, pages 75–80, 2006.

[Ott08] Jörg Ott. Delay Tolerance and the Future Internet. In *Proceedings of the 11th International Symposium on Wireless Personal Multimedia Communications*, 2008.

[OWS+06] Jörg Ott, Stephan Wenger, Noriyuki Sato, Carsten Burmeister, and Jose Rey. Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-based Feedback (RTP/AVPF). RFC 4585, July 2006.

[OX07] Jörg Ott and Lu Xiaojun. Disconnection Tolerance for SIP-based Real-time Media Sessions. In *Proceedings of the International Conference on Mobile Ubiquitous Multimedia (MUM)*, pages 14–23, 2007.

[Per07] Colin Perkins. RTP and the Datagram Congestion Control Protocol. Internet Draft draft-ietf-dccp-rtp-07.txt, Work in progress, June 2007.

[PHH98] Colin Perkins, Orion Hodson, and Vicky Hardman. A survey of packet loss recovery techniques for streaming audio. *IEEE Network*, pages 40–48, Sep/Oct 1998.

[PKH+97] C. Perkins, I. Kouvelas, O. Hodson, V. Hardman, M. Handley, J.C. Bolot, A. Vega-Garcia, and S. Fosse-Parisis. RTP Payload for Redundant Audio Data. RFC 2198, September 1997.

[RLM+06] J. Rey, D. Leon, A. Miyazaki, V. Varsa, and R. Hakenberg. RTP Retransmission Payload Format. RFC 4588, July 2006.

[SCFJ03] Henning Schulzrine, Stephen Casner, Ron Frederick, and Van Jacobson. *RTP: A Transport Protocol for Real-Time Applications*, July 2003. RFC 3550.

[Ste07] R. Stewart. Stream Control Transmission Protocol. RFC 4960, September 2007.

[SWB01] J.W. Seo, S.J. Woo, and K.S. Bae. A study on the application of an AMR speech codec to VoIP. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 3, pages 1373–1376, 2001.

[ÁHI08] Árpád Huszák and Sándor Imre. DCCP-based Multiple Retransmission Technique for Multimedia Streaming. In *Proceedings of the 6th ACM International Conference on Advances in Mobile Computing and Multimedia (MoMM)*, pages 21–28, 2008.