QoS Experiments with 802.11b For the Ipaq 3975
Status Report - 11/5/2002
CSC714 Term Project
Group 11

Outline

1. Preliminary Investigation and Results
2. Issues/Concerns
3. Proposed Schedule

1. Preliminary Investigation and Results:

All of us : Get familiar with the device and the development environment

Guru : Wrote/tested small initial groundbreaking applications on eVC++.

- Socket application using windows sockets.
- Sample applications to test basic UI features, unicode features etc.
- Maintain the project webpage and document progress.

Vasanth : Look for open source audio/video decoding engines for WinCE

- MpegTv is an open-source mpeg decoding engine with support for Win CE.
- The SDK provides both the source and libraries for the decoding and rendering threads that any streaming application needs.
- There is not much documentation about the API, but from responses in discussion groups, its an intuitive and reliable API.
- This has to be verified though. We plan to use this SDK for the client application, which is going to run on the Ipaq.
  Sample applications provided with the official distribution compile and work after some effort. The good news is that the
  performance of the engine is good, without any run-time concerns.

- Windows Media Encoder is a Microsoft provided SDK with which one can implement Streaming Servers.
  The efficacy of using this for implementing our streaming server is being investigated.

Anita : Researched the Real-time Transport Protocol.

- RTP is the Internet standard protocol for the transport of real-time data such as video and audio.
- It can be implemented on top of standard UDP services over unicast/multicast networks.

- It provides payload type identification, sequence numbering, timestamping and delivery monitoring services.
- RTP works in conjunction with the real-time control protocol (RTCP). RTCP provides control over delivery and quality of data.

2. Issues/Concerns:

Corresponding to step (1) of our proposal, we are thinking of two different approaches.

a) Instead of writing streaming client and server applications from scratch use already available media decoding/playing engines
and concentrate on implementing a RTP streaming layer on top of the engine.

b) In the case that the SDK proves to be cumbersome/ unreliable, write our own media decoding engine (this will be kept simple,
audio-only, pcm) and write our own streaming thread on top of it. This would involve us writing our own media player. This may divert us from our original goal of evaluating QOS.

We are still debating going for (a) or (b). Resolving this is our immediate concern.

3. Proposed Schedule:

Steps:

1. Develop the media decoding/rendering application. Use either (a) or (b).
   Vasanth & Anita : 11/17

2. Develop a RTP layer. Involves writing RTP server/client.
   Guru : 11/17

3. Insert hooks in both 1,2 to retrieve QOS metrics
   All of us : 11/22

4. Develop Applications simulating sporadic/aperiodic load.
   All of us : 11/22

5. Simulations and Measurements
   All of us : 11/25

   Within this there will be further work split based on different possible simulation scenarios.

6. (Open-ended) Investigate Middleware solutions to improve things
   TBD (depending on progress in steps 1-5)

7. Project Report and Final Demo
   All of us : 12/dd