

Hardware and Software Architecture of a Packet Telephony Appliance

Cormac J. Sreenan

AT&T Labs Research
Florham Park, NJ
cjs@research.att.com
<http://www.research.att.com/~cjs>

Collaboration with AT&T Labs colleagues:

***Mike Chan, Chuck Cranor, R. Gopalakrishnan,
Peter Onufryk, Larry Ruedisueli, Eric Wagner***

<http://www.research.att.com/~cjs/tops-project>



Talk Outline

Packet telephony

- Background and previous work

Telephone appliance

- Design issues and principles

Hardware architecture

- Networked processor
- Telephone design

Software architecture

- Communications mechanisms
- Telephone application components

Conclusion

- Status, future work, related work

Packet Telephony Background

Using packet networks for voice communication

- ❑ Driven by potential for cost savings and new services
- ❑ Today mainly based on PSTN dial-in to gateway
- ❑ Gateways interconnected over IP networks

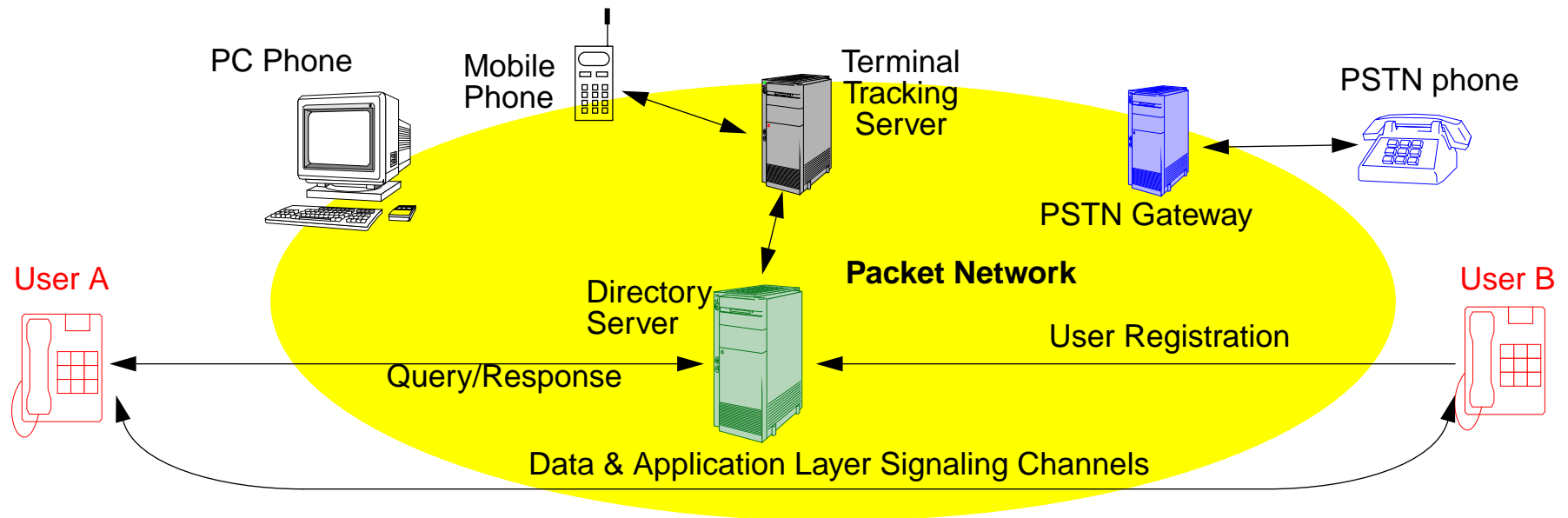
Need to examine consumer endpoint appliances

- ❑ Use of office LANs for voice and data
- ❑ Penetration of home access and in-home networks

Intelligence at both the end-points and in the network

- ❑ Opportunity to re-partition functionality
- ❑ Application specific signaling supports telephony features (conferencing, teleporting, etc.)
- ❑ Flexibility to support new services and capabilities (e.g. multiple media)
- ❑ Implications for endpoint design

Previous work: TOPS Architecture (NOSSDAV'98)



Call Flow:

- ❑ User/Terminal Registration with Directory Server
- ❑ Directory Query (returns a call handling profile)
- ❑ Application Layer Signaling (ALS) establishes calls

Other servers: PSTN gateway, Terminal Tracking Server

Why Not a PC based Packet Phone?

Too Expensive (\$500 too much for a phone)

Too Complex

- To use
- To install and configure
- To administer

Too Unsightly

- Big (keyboard + mouse + monitor + case)
- Loud (power supply fan, CPU fan, and disk)

Too Unreliable

- Never on when you need it
- Crashes often

Design Principles

Low cost (less than \$100)

- ❑ A consumer device (a phone to put in your bedroom)

Extensibility

- ❑ Packet telephony is in its infancy -> standards changing
- ❑ Research areas still exist: QoS, security, privacy, billing
- ❑ Must be able to support new/advanced services

Ease of Use

- ❑ Designed for people with no technical background
- ❑ Ordinary people unwilling to invest time to set-up, configure, and maintain complex devices

Reliability

- ❑ Always on and always works

Euphony ATM Telephone (EAT)



Features:

- **ATM-25 Network Connection**
- **RS232 (PC style DB9)**
- **Traditional telephone interface**
- **Case speaker and microphone**
- **External audio**
 - **Line output**
 - **Line / microphone input**
- **Two extra push buttons**
- **Three green status LEDs**
- **One red status LED**

Why Not Include More “Stuff” with the Phone?

Things we could have added

- ❑ LCD display (touch screen), more keys, keyboard, mouse,

Adding “stuff” is bad

- ❑ Makes phone expensive and complex
- ❑ Would anyone really use it?
- ❑ How many ISDN phone features do you use?

Our approach for advanced features is a soft interface

- ❑ Phone runs web server
- ❑ Advanced features available via remote browser and other network devices
- ❑ You pay only for those features you want

Euphony Networked Processor

Like an Intel 486 DX2 66 with: signal processing instructions, audio interface, network interface, and system logic all for about \$7

RISC Processor

- ❑ R4000 like (MIPS II) RISC processor

DSP Instructions

- ❑ Pipelined multiplier, Extract/Saturate instructions

ATM Interface

- ❑ single ATM Forum UTOPIA, single or dual AT&T DPI

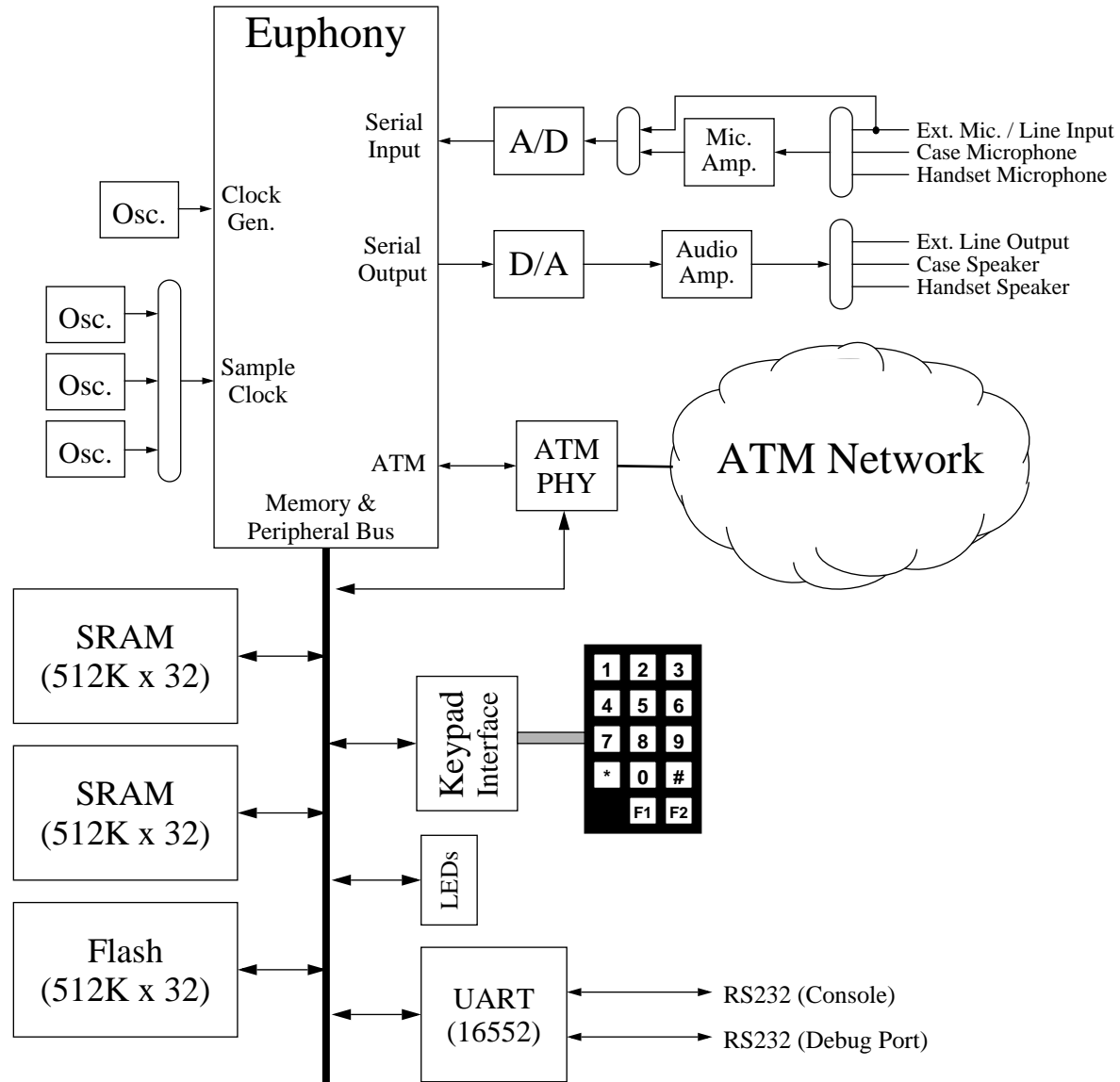
Digital Audio Interface

- ❑ Serial interface for D/A and A/D

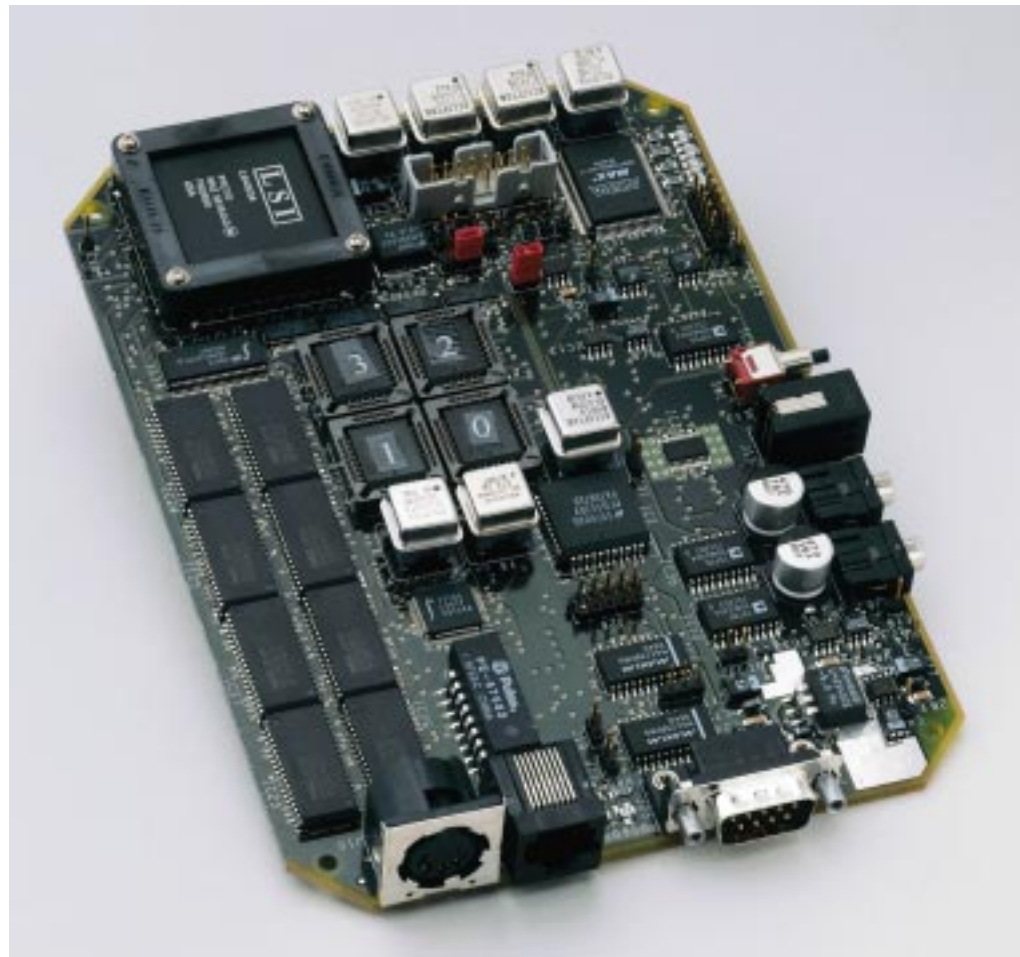
Support Logic

Low Power (500 mW)

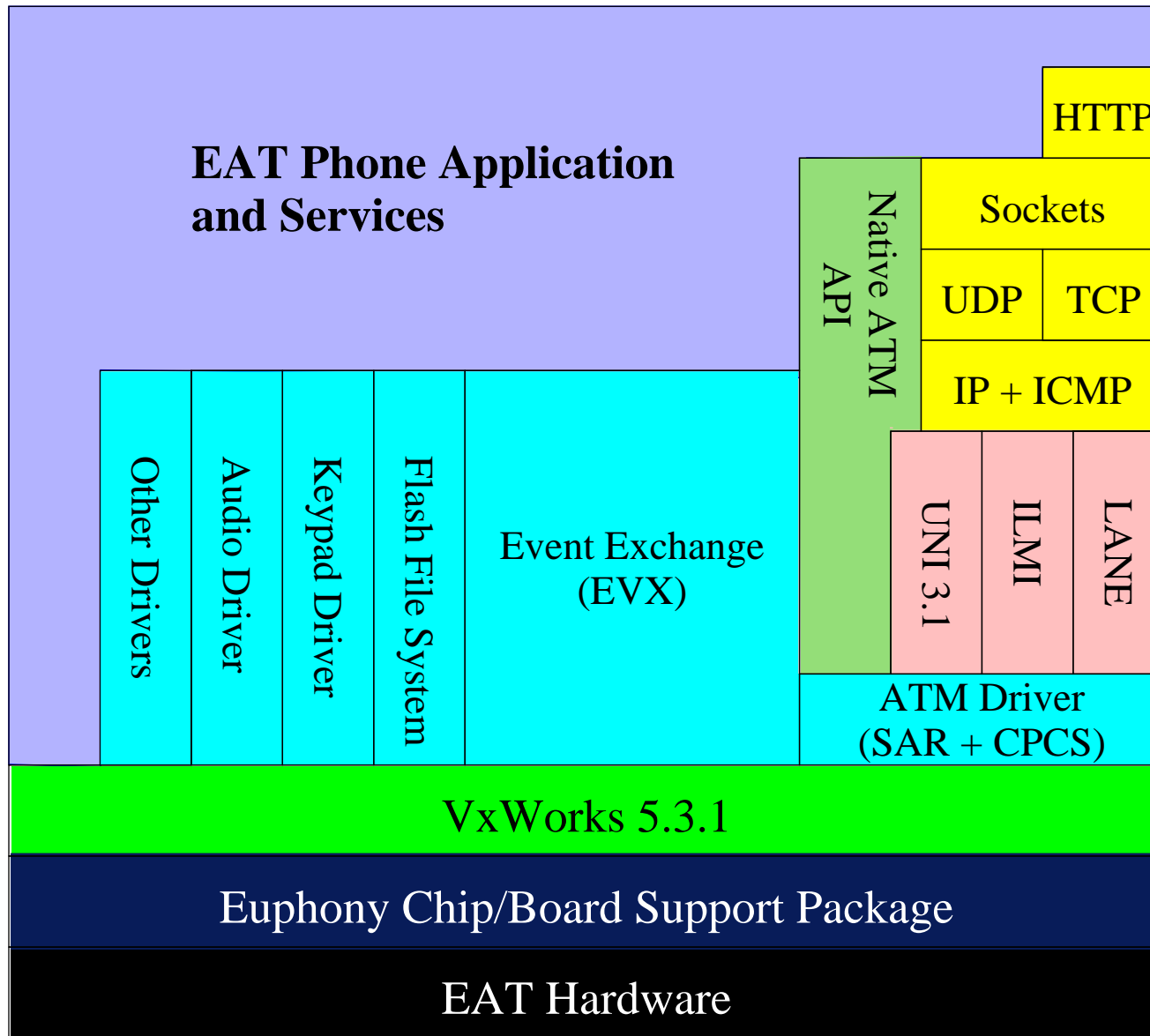
Euphony ATM Telephone (EAT) Block Diagram



EAT System Logic Board



EAT Software Architecture



Why an RTOS?

Why do you need an OS at all?

- ❑ Packet telephony application quite complex
- ❑ Need tasks and interrupt handlers
- ❑ Need standard libraries

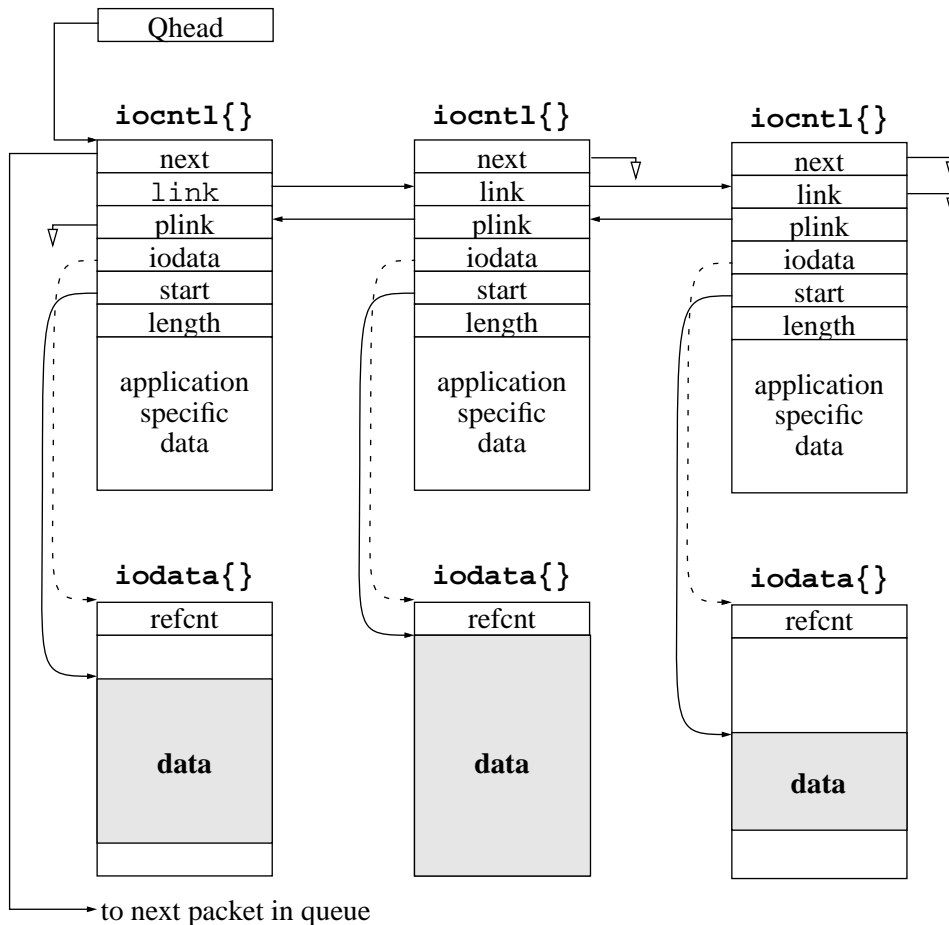
Why not Linux?

- ❑ Timesharing environment brings too much baggage
- ❑ Hard to scale down, Not real-time
- ❑ Poor embedded development tools

Why VxWorks (could be any RTOS)

- ❑ Small and efficient real-time kernel
- ❑ Scalable (minimum kernel ~64K-bytes)
- ❑ Ported software (web server, SNMP, Java, etc....)
- ❑ Excellent cross development tools

IObufs



- Copy reduction techniques used to reduce memory & latency
- Similar to BSD Unix mbufs
- *IObufs* are not mbufs
 - Allows application specific info
 - Separate control and data blocks
 - Data block can be of any size
- *IObufs* provide a uniform buffering mechanism used by all modules
 - I/O (e.g., network and audio)
 - Application

Intra-Appliance Communication

Initial software implementation was too unstructured

- ❑ Tightly integrated, hard to debug and add new features
- ❑ Motivation for a modular design that allows evolution and experimentation
- ❑ Avoid tight coupling between appliance functions
- ❑ Flexible communication for media buffers and events
- ❑ Example events: call states, on/off hook, key presses

Efficient audio movement using zero-copy IObufs

- ❑ Coupled with Event eXchange (EVX) for distribution
- ❑ One-to-many, flow controlled, sender/receiver decoupling

Examples of EVX use

- ❑ Key presses sent to digit collector and tone generator
- ❑ Hook events potentially of interest to several modules

EVX

EVX delivers events posted on a “sending” port to one or more interested modules on their “receive” ports

- ❑ Module communication defined in terms of port names and event types
- ❑ Events can be delivered to multiple receivers
- ❑ Sender does not need to know about receivers
- ❑ Data delivered in IObufs using reference counts
- ❑ Flow control to prevent overrun

EVX API for creating ports, sending/receiving events, waiting for events and network I/O

EVX application consists of three parts

- ❑ Application modules
- ❑ Initialization code that configures EVX connections
- ❑ EVX API library + EVX thread for event processing

EAT Software

I/O

- Audio, network
- Keypad
- Hook monitor

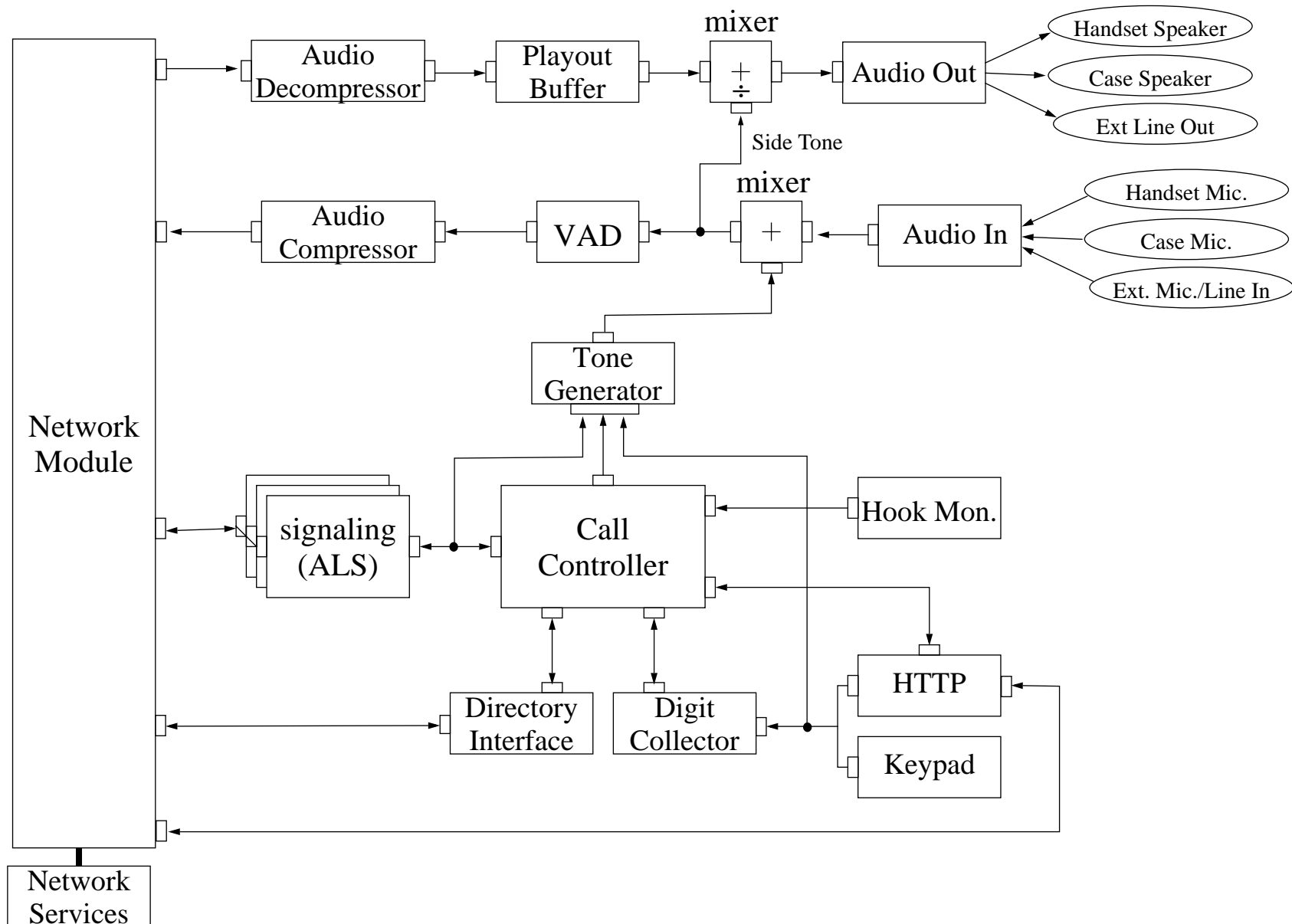
Audio path

- Compression/Decompression
- Voice Activity Detection (VAD), Playout buffering
- Mixing audio samples

Telephony

- Tone generator
- Signaling, Directory access, Call controller
- Digit collection

EAT Modules and Data Flow



Status and Future Work

Ongoing

- ❑ Deploying more than 20 EATs in offices
- ❑ Building a T-1/PBX gateway using ALS

EVX

- ❑ Dynamic configuration of new services, coders etc

Signaling

- ❑ Use of per-call choice of signaling protocol

Advanced Services

- ❑ What features should be in network servers and what features should be implemented by “intelligent” devices?
- ❑ High quality music end-point
- ❑ Voice enabled user interface
- ❑ Interactions with network services

Related Work

Packet Telephony Directories and Call Signaling

- ❑ IETF Session Initiation Protocol (SIP), ITU H.323, etc

Packet voice

- ❑ Early voice networking, e.g. Etherphone, ISLAND

IP Telephony appliances

- ❑ Off-the-shelf (e.g. Selsius, Symbol)
- ❑ Fixed choice of coding and signaling (H.323)
- ❑ Services via LAN-based PBX PC

Communication

- ❑ Zero-copy techniques, Fbufs, Rbufs, etc
- ❑ Distributed event services
- ❑ Conference bus protocols, Message/software buses

Conclusion

Comprehensive design and implementation of a packet telephony appliance

- ❑ Low-cost, simple telephony device
- ❑ Easy to use and reliable
- ❑ Suitable for experimentation

Hardware

- ❑ Networked processor with ATM and digital audio interfaces
- ❑ Traditional styling, 12-button keypad, handset

Software

- ❑ Real-Time OS, ATM & IP protocol stacks
- ❑ Efficient and extensible: zero-copy IObufs and EVX
- ❑ Advanced features and control using a web browser