PSTN – SIP Telefon Gateway

Wenn man von Rauchzeichen absieht, gibt es folgende Arten von Kommunikationsystemen:

Festtelefon..... ATM-, PSTN-Netzt
Mobiltelefon..... Mobilfunknetz
VoIP Telefonie..... Internet
E-Mail ..... Internet
SMS..... Mobilfunknetz
WWW ..... Internet
Voice-Mail (Anrufbeantworter)
Video Telefonie

Im Zuge der technologischen Entwicklung und aus Gründen der Produktivitätssteigerung und Kostenersparnis ist man bestrebt, die zuvor genannten Systeme zusammenzuführen. Hieraus ergibt sich eine neue Generation von multimediafähigen Kommunikationsgeräten. Obwohl dies kein leichtes Unterfangen ist, gibt es bereits solche Geräte: Mobilfunkgeräte von Nokia und Samsung mit WLAN-Interface; Video-Telefonie mit UMTS etc.

Mit diesem Dokument werde ich versuchen, die schon gemachten ersten Schritte zu verstärken und mehr akzentuieren, damit hier das Profil deutlich wird. So wie dies mit der SIP-Telefonie bereits einen Anfang gefunden hat.

In der Infrastruktur der kleinen und mittleren Unternehmen existieren in der Regel zwei voneinander unabhängige, getrennte Systeme:

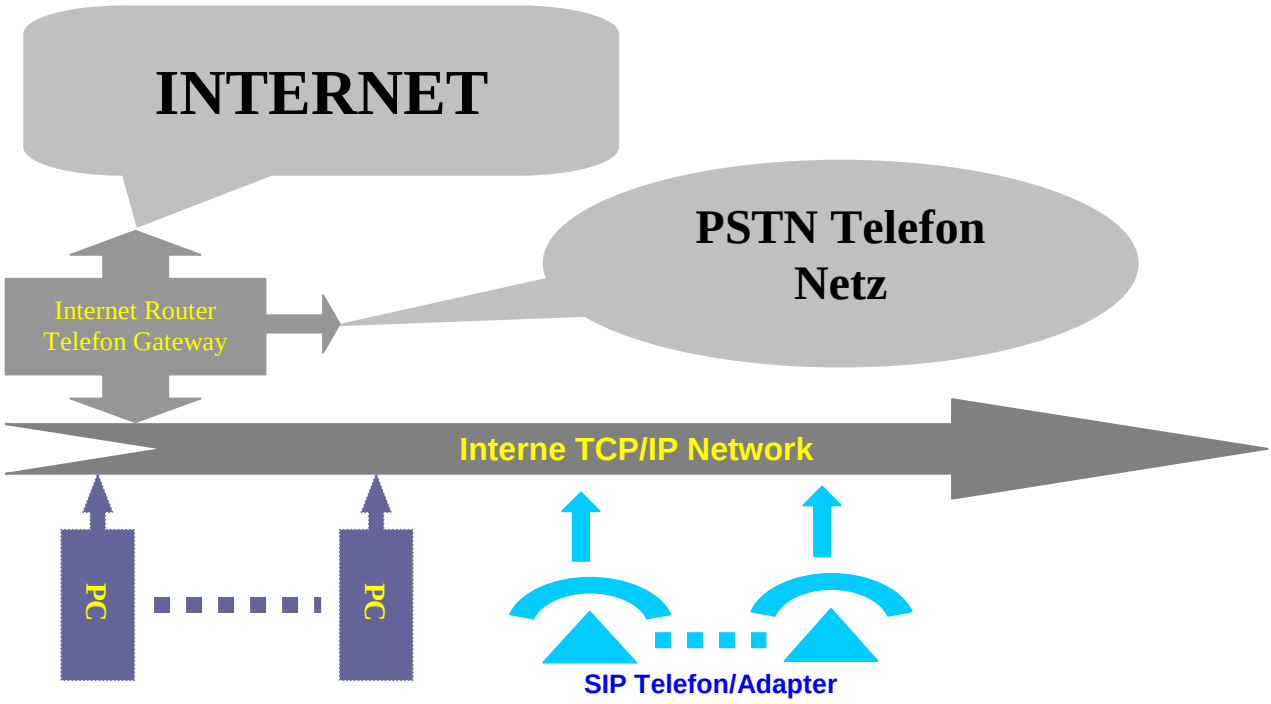
- Telefonnetzt (ISDN, Legacy)
- Computer-Netz

Diese führen in friedlicher Koexistenz ein Dasein: jeweils getrennte eigene Vernetzung, eigene Anschlussdosen, eigene Installation und eigene Administration.

Mit SIP können die beiden Netze zusammengeführt werden. Hieraus ergeben sich Spareffekt wie z.B. bei der Vernetzung und Administration aber auch neue flexible Einsatzmöglichkeiten und Produktivitätssteigerung durch netzweite Unternehmenstelefonie sowie Umgehung von Medienbrüchen. Ausserdem gelten die hohen Sicherheitsstandards der vorhandenen Netzstruktur auch für die Telefonie.

Das TCP/IP-Netz ist ein internes, qualitativ hochwertiges Kommunikationsmedium für Daten und Sprache. Es ist also prädestiniert für den Einsatz der VoIP-Telefonie in Unternehmen als internes Kommunikationsmedium. Extern kann nach wie vor konventionell über PSTN kommuniziert werden. Nur ausgewählte Strecken werden Schritt für Schritt durch SIP ersetzt. Somit bleiben auch die bisherigen Investitionen geschützt.

Die Struktur des Kommunikationssystems in Unternehmen.



Diese Architektur bietet nicht nur Kostenersparnisse. Auf diesem Wege es ist auch möglich, die ansonsten fest vergebenen Nebenstellennummern, vollständig dynamisch zu verwalten und zwar intern wie extern. Man kann SIP-Registrierungsmechanismen/Locationservice z.B. für Gesprächsweiterleitung auch für normale Telefonate nutzen.

Die Schlüsselrolle in dieser Empfehlung/Infrastruktur kommt auf **Internet-Router/Telefon-Gateway**.

Nun für die Realisierung des empfohlenen Kommunikationsystems müsste ein Internet-Router alle unten aufgeführten Sub-Systeme unter einem Dach zusammenfassen:

- **SIP to PSTN Telefon-Gateway**
- NAT und Firewall
- WEB-Proxy und WEB-Server als Admin.-Frontend
- E-Mail Proxy und -wahlweise - E-Mail server
- Voice Mail und Anrufbeantworter
- Autorisierungs- und Autentifizierungs-Server
- DNS, DHCP und XNTP
- VPN Service

Jedes dieser Sub-Systeme ist für sich ein technisch anspruchvolles Gebiet, dessen eigene Entwicklung mehrere Jahre in Anspruch nehmen würde. LINUX-Plattform -als Open Source- bietet hier die einzig sinnvolle Lösung:

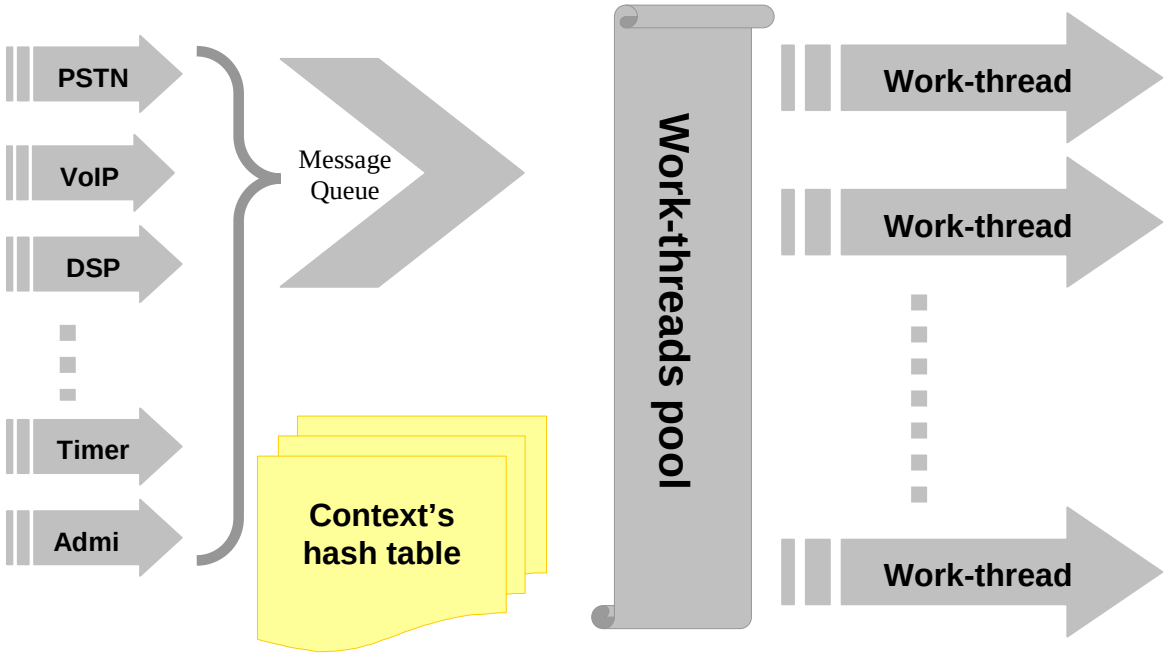
- **SIP to PSTN Telefon-Gateway – Eigene Entwicklung**
- NAT und Firewall – ist Bestandteil vom LINUX Kernel
- WEB-Proxy und WEB-Server – Squid und grosse Auswahl von WEB-Servern
- E-Mail Proxy und -wahlweise - E-Mail server – EXIM4 oder andere E-Mail System
- Voice Mail und Anrufbeantworter – fertig implementiert als Teil isdn4linux
- Autorisierungs- und Autentifizierungs-Server – RADIUS, SSL und SSH
- DNS, DHCP und XNTP – Standardaufgaben für LINUX-Rechner
- VPN Service – große Auswahl von als „gut“ getesteten VPN Paketen

Alle Sub-Systeme existieren bereits. Und LINUX ist bekanntlich ein zuverlässiges und stabiles System.

Es gibt aber einen Haken! Die teilweise sehr umfangreichen Sub-Systeme müssen integriert werden, um ein kommerzielles Produkt zu erzeugen. Solche Integrationsprobleme (Installation und Administration) werden bei typischen LINUX-Installation mit Hilfe der Skriptsprachen PERL und/oder PYTON gelöst.

Der Unterschied zwischen der typischen LINUX-Installation und dem oben skizzierten Internet-Gateway besteht darin, dass die Hardwarevoraussetzung rudimentär ist. Aus Preisgründen muss es mit langsamen Prozessoren und wenig Speicher auskommen! Genau dies stellt besondere Anforderungen an die Softwareauswahl. Diese müssten z.B. die Wiederverwendbarkeit der von Softwarepaketen benutzten Bibliotheken optimieren.

SIP to PSTN-Gateway macht aus dem LINUX-Rechner eine Telefon-Vermittlungsstelle (Telefon-Switch). Der **SIP to PSTN-Gateway** ist als Multithreaded-Single-Event-Queue-Applikation mit folgender Thread-Struktur implementiert:



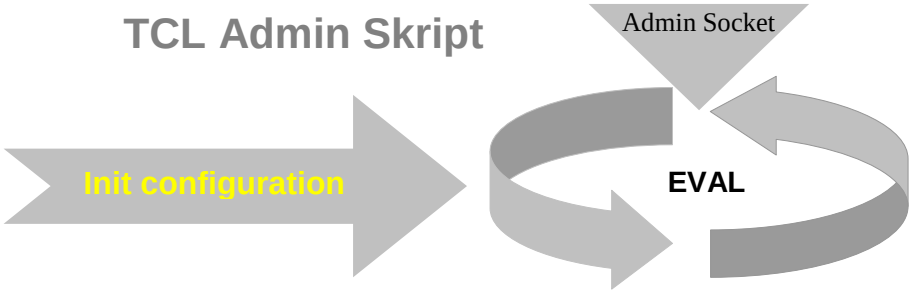
Der linke Teil des Bildes zeigt die Events-Threads. Sie stellen das jeweilige Kommunikations-Interface mit spezifischer Hardware dar. Anliegende Hardware Signale werden zuerst ins Common-Applikation-Event-Format konvertiert und dann in die Applikations-Message-Queue gestellt. Der rechte Teil zeigt den Pool der zu bearbeitenden Threads (Work-thread). Work-thread selbst hat keinen Kode zur Events-Bearbeitung, alle Events sind Kontext bezogen. Kontext Objekt/Class hat alle relevanten Daten und Methoden, die für die Event-Bearbeitung notwendig sind. Mit anderen Worten: Work-thread ist eine allgemeine Engine, die ein Event aus der Message-Queue holt, den passenden Kontext im globalen Kontext's-Hash-Table findet und schliesslich die kontextinternen Methoden für die Event Bearbeitung verwendet .

Diese Architektur hat folgende Vorteile: einfache Skalierung von Applikationen (die Menge vom Work-thread ist ein Konfigurations-Parameter); beliebig erweiterterbar auf neue Anwendungsgebiete (z.B. neue Kontext-Typen); keine Änderung der allgemeinen Applikations-Architektur. Diese Architektur ist flexibel im Umgang mit Systemressourcen (optimale Nutzung von Multiprozessor-Systemen). Neue Hardware mit neuem Events-Thread ist einfach zu integrieren.

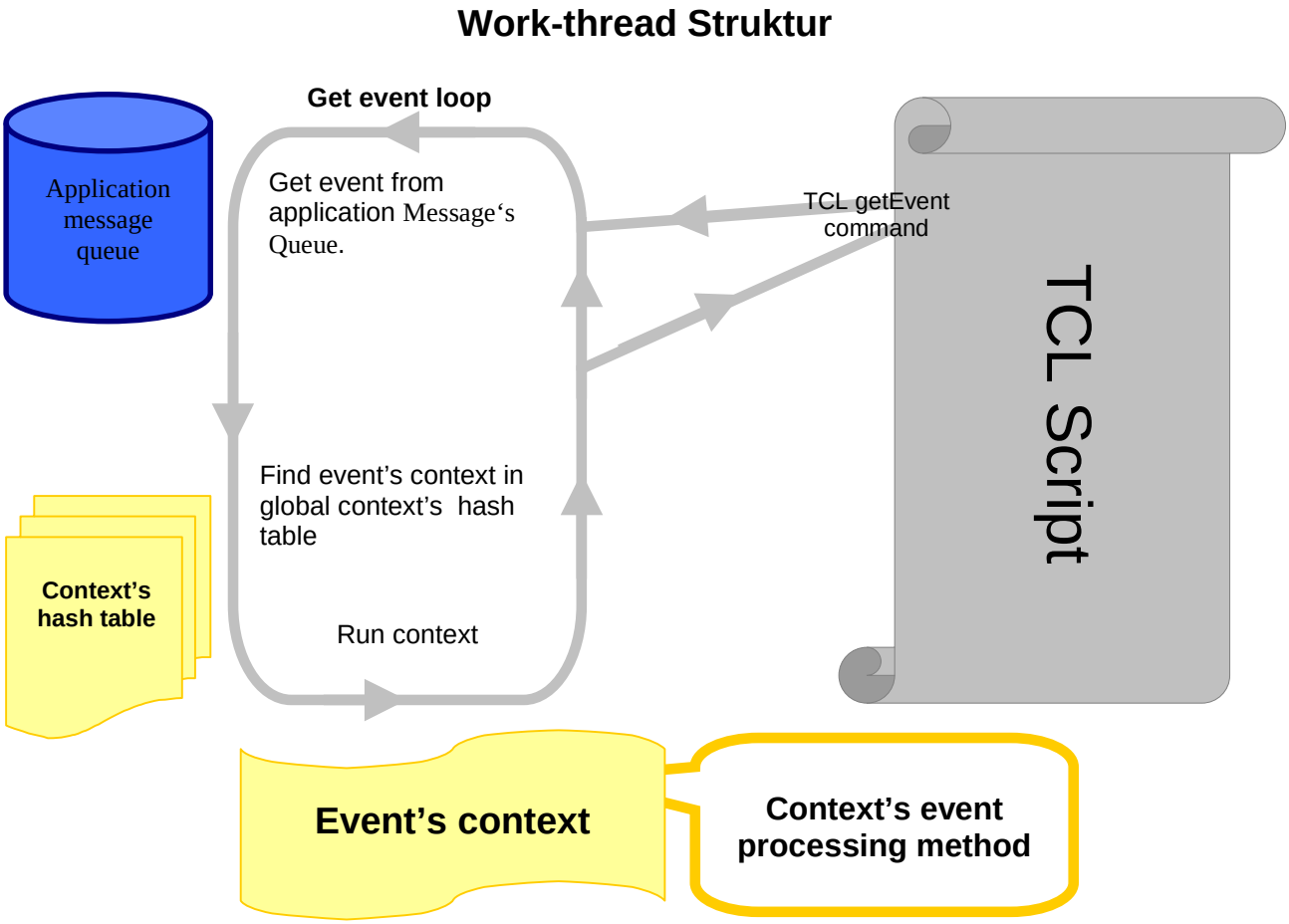
Nun folgt eine kurze Beschreibung der einzelnen Events-Threads:

- PSTN – konvertiert Telefon-Events (Incomming Call, Ringing etc.) ins Applikations-Event.
- VoIP – wartet auf IP-Socket-Events (z.B. SIP-Messages) und generiert entsprechende Applikations-Events.
- DSP – wartet auf Tone-Detektion-Events (DTMF-, FAX- und MODEM-Tone) und generiert entsprechende Applikations-Events. Auch handelt Events um Aussagen zu produzieren.
- Timer – generiert applikationsweite Timer-Events. Timer-Events sind auch kontextbezogen.

- Admin – dieses Thread ist eigentlich Applikation „main(...)“ Funktion. Die Applikation „main(...)“ startet ein TCL Interpreter, der eine Konfiguration/Administrations TCL-Skript ausführt. TCL (Tool Command Language) selbst ist eine von John K. Ousterhout (Berkeley University, USA) entwickelte Skriptsprache, die zum einen einfach mit neue Befehle (in C/C++ Implementiert) erweitert kann und zum anderen anstelle von PERL verwendet werden kann. Diese Eigenschaften machen TCL zu einer idealen applikationsinternen Skriptsprache für embeded Systeme. Das Admin-Skript erstellt mit Hilfe der Anwender-Befehle eine Konfiguration und startet dann eine Lisening-Schleife, die auf Befehle auf Admin-Socket wartet.



Im folgenden Bild wird der Work-thread-Aufbau im Detail dargestellt. Allgemein: Work-thread ist nichts Anderes als ein TCL Interpreter. Zu Anfang startet Work-thread ein bereits konfiguriertes TCL-Skript und der Rest wird eben in diesem Skript erledigt. Das Skript selbst ist eine Schleife, die mit dem in unserer Applikation implementierten TCL-Befehl „getEvent“ gestartet wird. Sie bekommt ein Event aus der Applikation-Message-Queue und startet die Event-Bearbeitung. Zur Event-Bearbeitung werden Anwender-Befehle verwendet. Diese wiederum rufen interne Kontext Methoden auf und gewähren Zugang zu Kontext-Daten.



Um die Event-Bearbeitung zu optimieren, bietet „getEvent“ zwei Bearbeitungswege. Erst wird ein Event aus der Applikations-Message-Queue geholt, dann Kontext geholt und Kontext‘s Event processing Methode aufgerufen. Der Returnwert vom „ProcessEvent“ entscheidet darüber, ob Event noch weiter in TCL bearbeitet wird und die Steuerung weiter an das TCL-Skript abgegeben werden muss oder Kontext‘s Event processing Methode schon alles

erledigt hat (Feste-Reaktion-Event). Dadurch kann unterschieden werden zwischen komplizierten (wo man in der Bearbeitung Flexibilität von TCL braucht) und einfachen wo die Reaktion auf ein Event fest definiert ist und als Kontext‘s Event processing Methode implementiert. Selbstverständlich kann TCL auch „Feste-Reaktion-Event“ bearbeiten, das aber kostet Zeit.

TCL übernimmt noch eine wichtige Funktion: Ein- und ausladen von dynamischen Bibliotheken (Shared Library).

In der oben beschriebenen Applikations-Struktur spielt TCL die zentrale Rolle und bringt folgende Vorteile:

1. Die Applikations-Konfiguration kann parametrisiert oder programmiert werden. Das z.B. ermöglicht dem Gerät, eigene Umgebung selbständig zu analysieren und dementsprechend eigene Konfiguration anzupassen. Oder serielle Installationen, wobei mit einem Skript mehrere Gateways installiert werden. Man kann später den Switch neu konfigurieren ohne Neustart.
2. Admin-Schnitstelle akzeptiert nicht nur einzelne Befehle. Auch komplizierte Skripte können ausgeführt werden.
3. Es steht jedem grafischen Frontend frei, die Admin-Schnittstelle von Aussen anzusprechen z.B.:
   vom WEB Server über CGI-Schnittstelle
   vom anderen TCL-Skript über SSH-Tunnel
   vom TK grafischen Frontend
4. Die Verwendung von TCL bietet die Möglichkeit, langsam und behutsam vom Prototyp zum Produkt zu migrieren. Zuerrst wird die Funktionalität als TCL da sein und danach werden einzelne Teile mit C/C++ Kode ersetzt.
5. Die Verwendung von TCL in Admin-Thread ermöglicht, auch andere Switch-Sub-Systeme effektiv zu konfigurieren und administrieren. Somit ist Admin-Thread das zentrale und einzige Systemmanagement-Portal.