GnuGk – The GNU Gatekeeper

OpenSource Telephony
Summit 2006

Jan Willamowius

2006-05-02

http://www.gnugk.org

Project Overview (1)

- started in 1999, GPL licence
- all regular H.323 gatekeeper features
  - address translation (alias to IP)
  - access control, call authorization, accounting
  - call routing
  - etc.
- support for Linux, Solaris, FreeBSD, MacOS X and Windows

Project Overview (2)

- authorization and accounting with various backend systems
  - plain text file
  - SQL
  - Radius
  - via TCP/IP
- gatekeeper clustering and failover support
  - child-parent gatekeepers
  - neighbor gatekeepers (interzone communication)
  - alternate gatekeepers
  - retry failed calls on another route

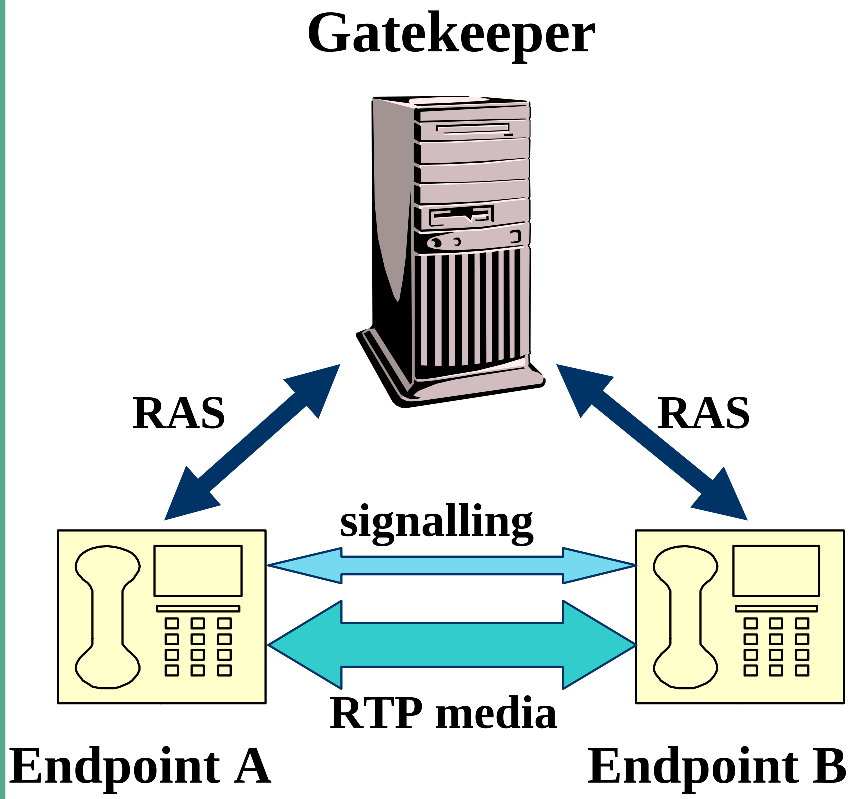
Project Overview (3)

- E.164 number rewritting
- NAT traversal - both outgoing and incoming calls
- support for various versions of H.323 protocol (V1 endpoints, some V4 features)
- H.235 security (authentication)
- CTI functions:
  - inbound call routing (“virtual queues”)
  - call transfer
- configuration changeable at runtime
- monitor and admin interface via TCP
- Graphical User Interface in Java for monitoring

Operational Modes

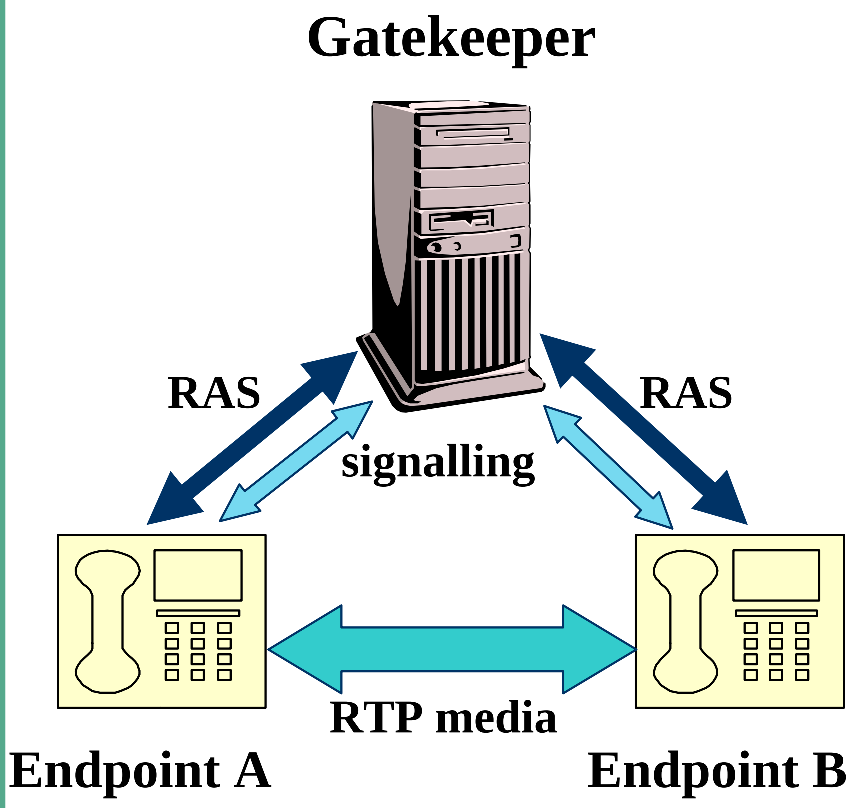
- direct signalling mode
- gatekeeper routed signalling mode
- full proxy mode (signalling + RTP media)

Direct Signalling Mode



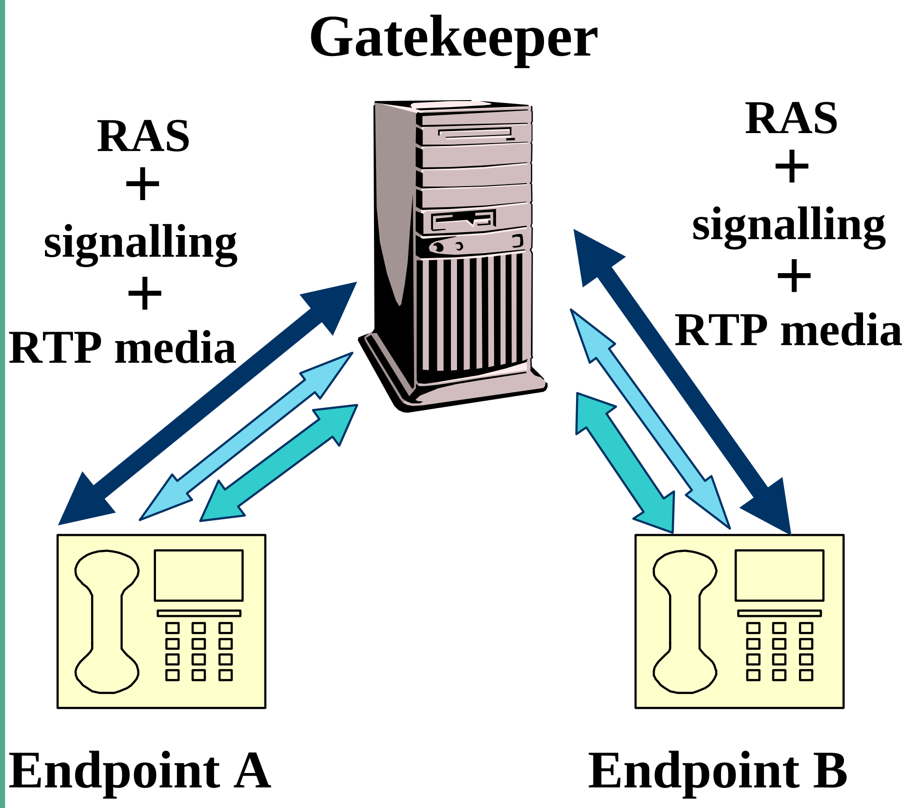
- only RAS channel between endpoints and the gatekeeper
- signalling directly between endpoints
- very good scalability
- lack of precise call control

Gatekeeper Routed Signalling



- signalling channel is routed though the gatekeeper
- precise call control (authorization, accounting)
- additional services like call transfer
- good balance between peformance and flexibility

Full Proxy Mode

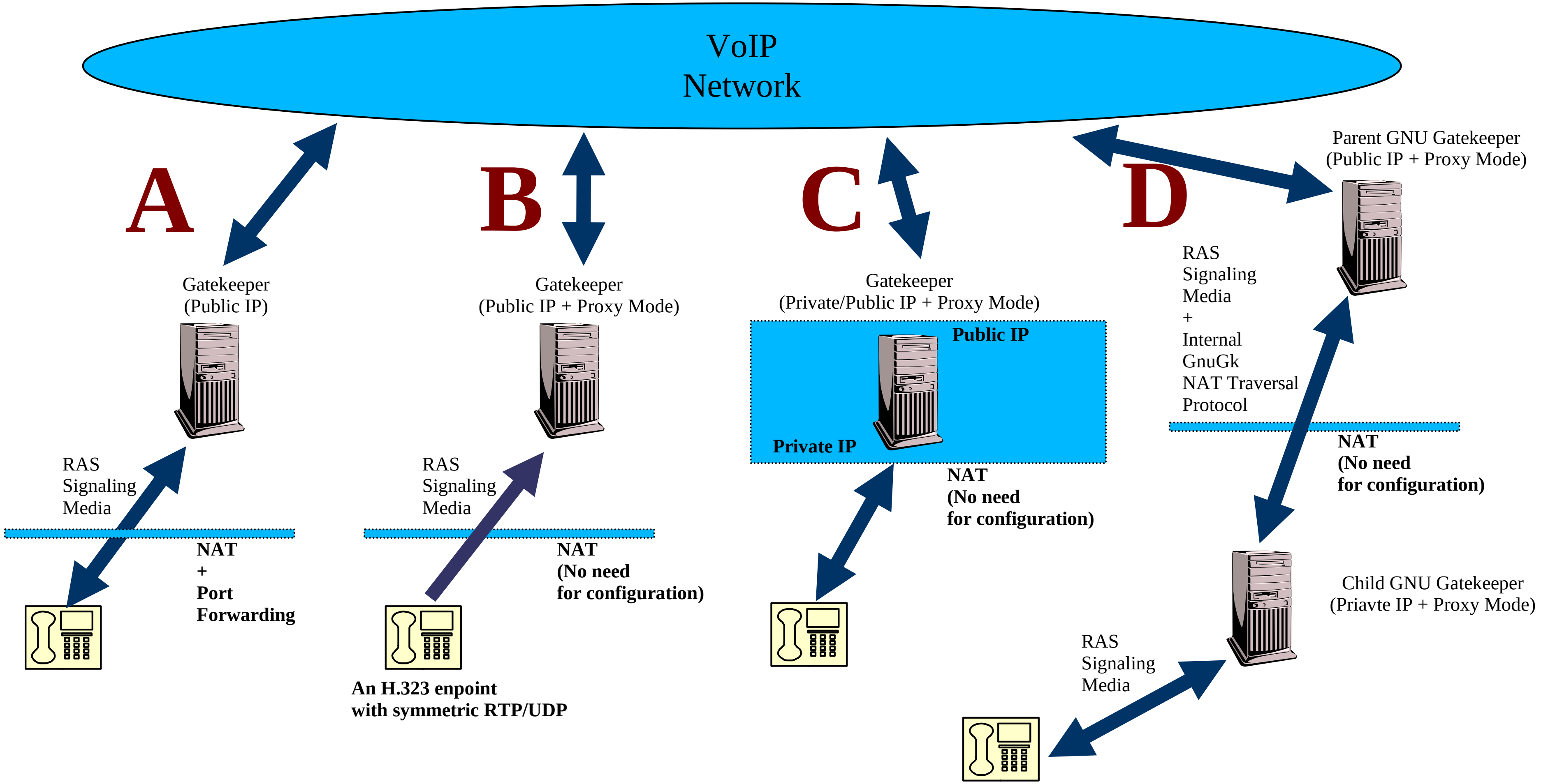


- all data (RTP audio, RTP video, T.120 data)
is routed though
the gatekeeper
- no direct communication between endpoints
- high CPU/bandwidth consumption
- designed to allow firewall/NAT traversal

NAT Traversal (1)

- 5 possible scenarios:
  - A)endpoint behind NAT, port forwarding enabled
  - B)(outbound calls only) endpoint behind NAT that knows how to use symmetric RTP UDP, gatekeeper in proxy mode
  - C)gatekeeper (proxy mode) on a NAT box (with access to both internal and external network interfaces
  - D)gatekeeper behind a NAT box, registered as a child with a parent GNU Gatekeeper (both have proxy mode enabled), both use internal NAT traversal protocol
  - E)endpoint behind NAT, knows how to use symmetric RTP/UDP and internal GnuGk NAT traversal protocol

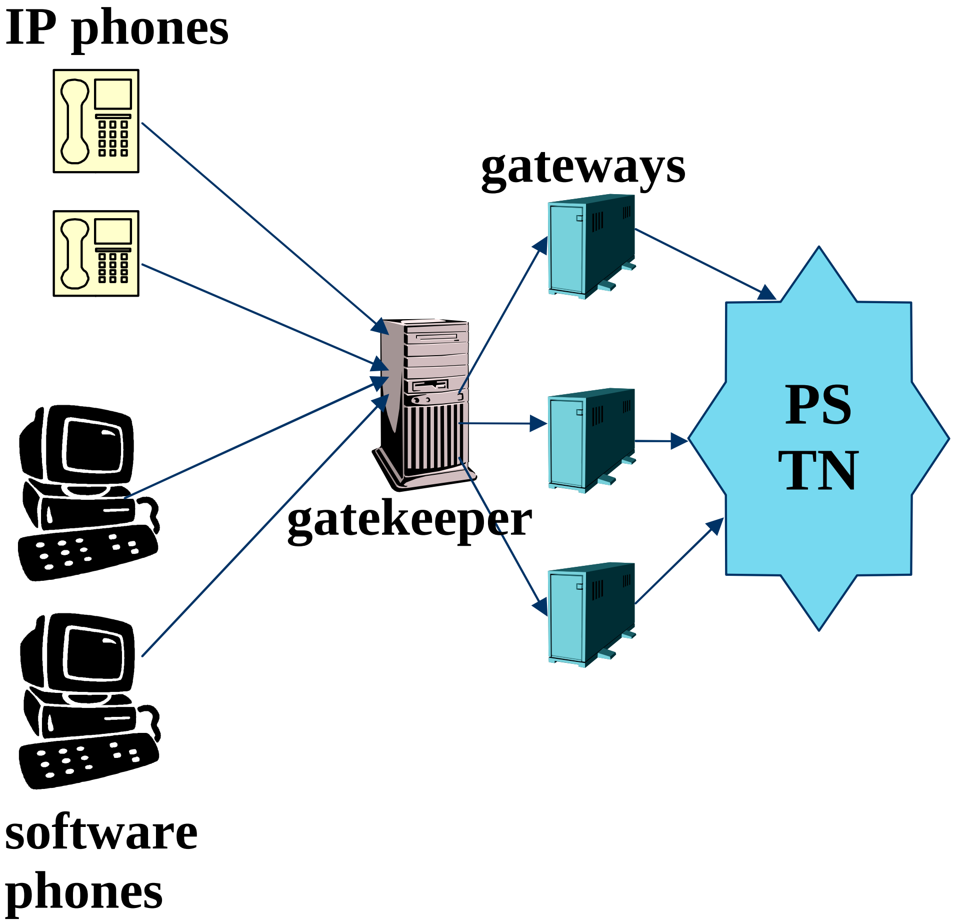
NAT Traversal (2)



GnuGk Deployment Scenarios

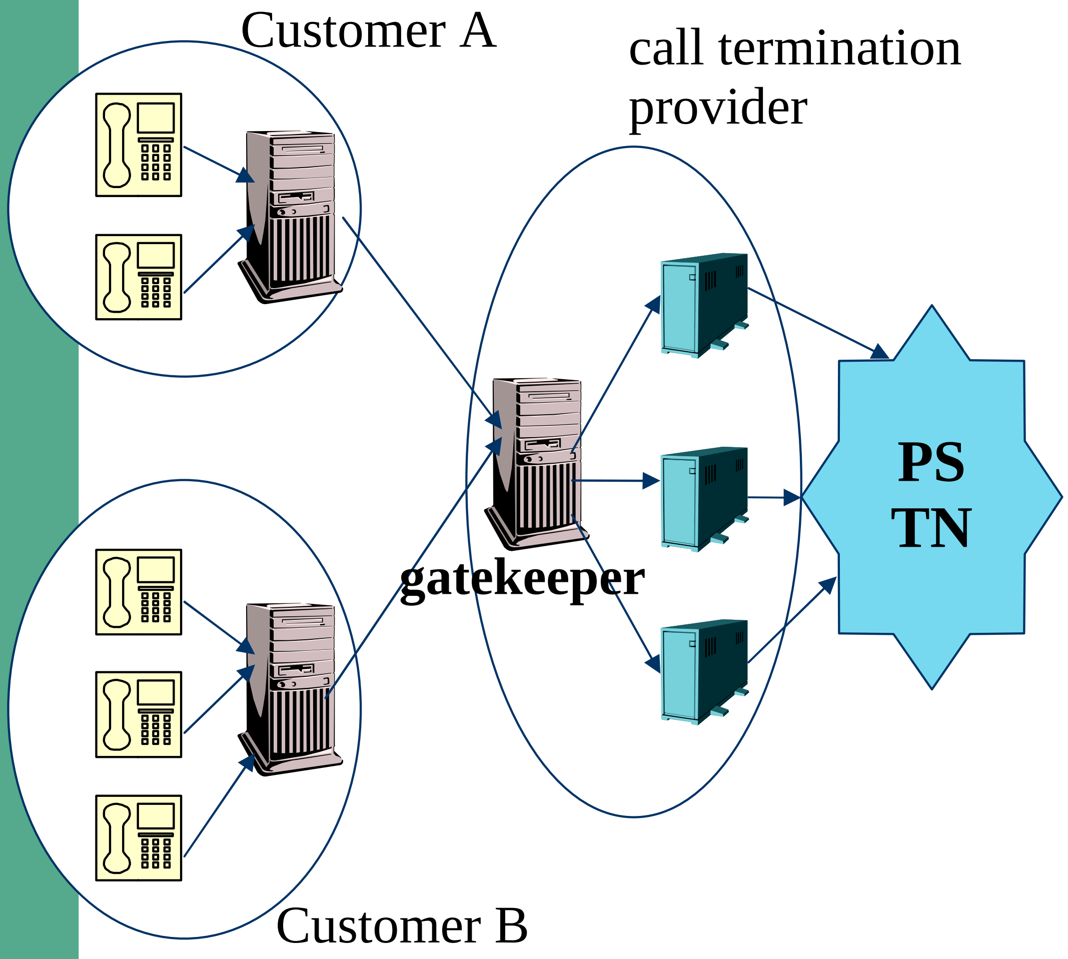
- Prepaid VoIP Telephony
- Call Termination Services
- VoIP Call Center
- PBX Replacement
- and much more ...

Prepaid Calling



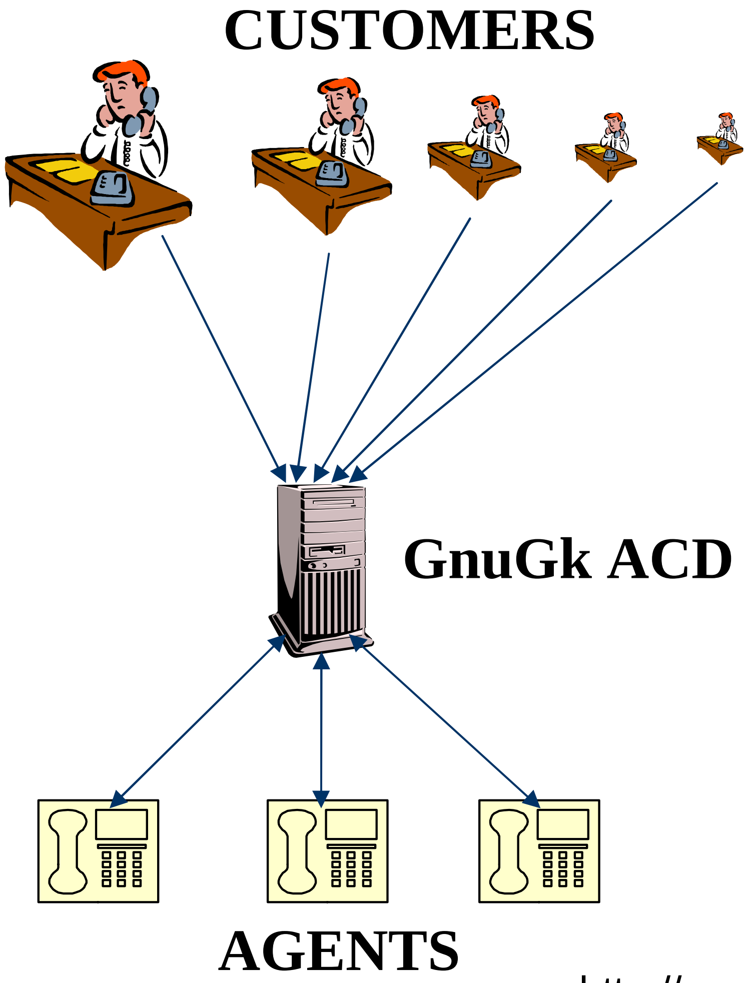
- call authorization and accounting
- enforcing limit on call duration
- easy integration with Radius and existing billing systems
- can be easily built from open source components only:
  - GnuGk + Radius server + SQL database

Call Termination Services



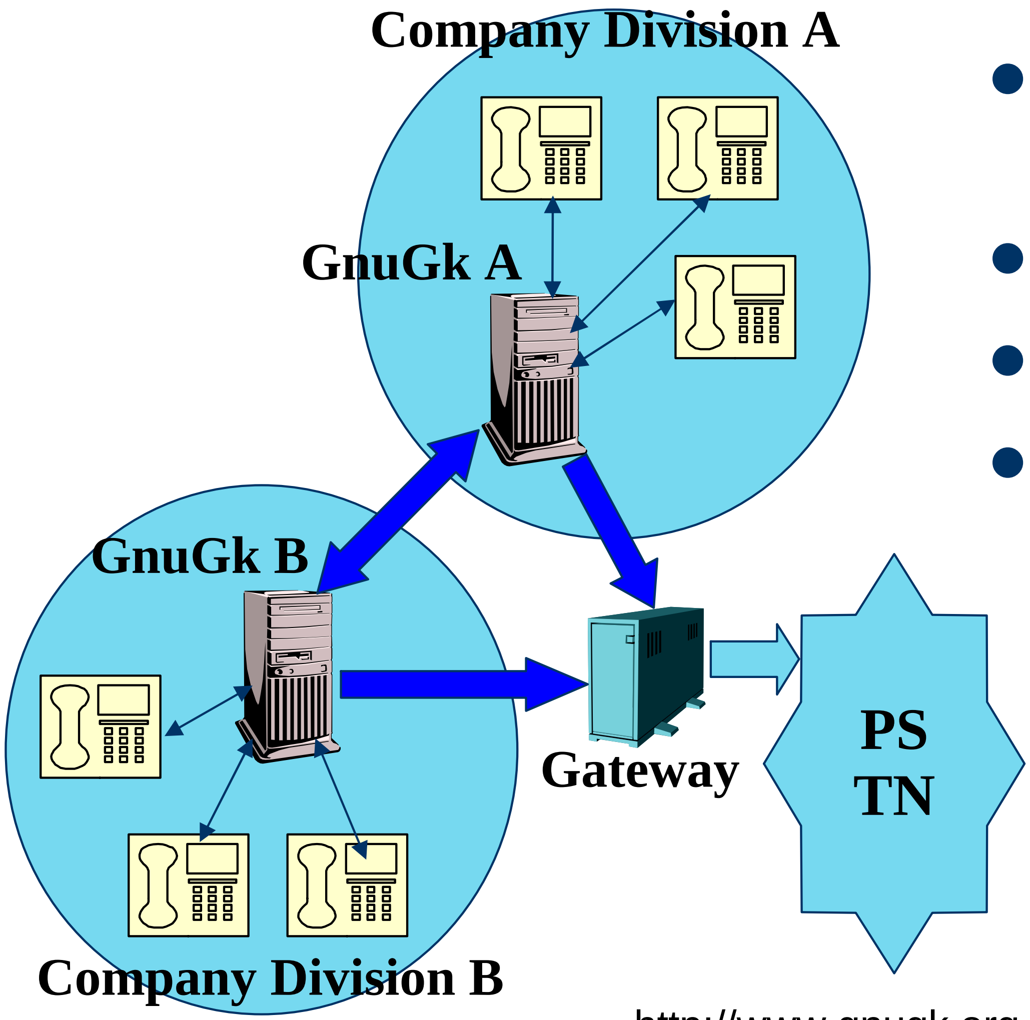
- call authorization and accounting (gatekeeper routed signalling mode)
- call routing decisions:
  - route the call to a specific gateways
  - route the call other call termination providers

Call Center



- ACD application (Automatic Call Distribution)
- calls to a single number are distributed to many agents (eg. hotline)
- various call distribution policies:
  - longest idle
  - first idle
  - round robin
  - TODO: skill based

PBX Replacement



- internal calls within the company
- inter-division calls
- numbering plans
- cheap PSTN calls

GnuGk Configuration

- manual in download archive (chapter 3 is a short tutorial)
- all configuration settings are read from a text file
  -
    ```
    [Gatekeeper::Main]
    Fourtytwo=42
    Name=GnuGk

    [RoutedMode]
    GKRouted=1

    [GkStatus::Auth]
    rule=allow
    ```
- can reload changed config at runtime

Accounting / Billing

- many acct modules
  - flat file (FileAcct)
  - Radius (RadAcct)
  - SQL (SQLAcct: PostgresSQL, MySQL)
  - Telnet interface (StatusAcct)
- SQL billing application for PostgresSQL in contrib/ directory
  - started as an example for an OSTS 2004 tutorial, now part of the GnuGk package
  - flexible billing **engine**

SqlBill (1)

- small but complete core for a billing/tariffing engine
- SqlBill provides
  - endpoint authentication by means of username/password, username/IP or IP only and alias control
  - endpoint/call authorization (allowed destinations, maximum call duration limit, account balance)
  - real-time account/call billing
  - support for prepaid/postpaid, originating/terminating account types
  - flexible tariffing engine

SqlBill (2)

- SqlBill does not provide
  - bussiness logic (invoicing, detailed customer data, payment processing, etc.)
  - user interface (minimalists can use pgAdmin;)
- technical details
  - can work on large databases
  - processes 50 calls / second on an average PC machine
  - communicates through RADIUS or directly with GnuGk
  - interfaces with PHP/.NET/ODBC applications easily
  - extendable to interoperate with other protocols/software

GnuGk Telnet Interface

- the „status port“
- interface for humans and external applications
- interface to non-GPL code
- remote administration
  - configuration changes/reloads
  - gatekkeper statistics (endpoints, total / active calls etc.)
  - manual call disconnect and endpoint unregistration
  - username/password based access authentication
  - call routing (“virtual queues”)
- live CDR output (new StatusAcct module)

Telnet Interface Applications

- Monitoring
  - Java GUI
  - GnuGk PHPStatus
- Call Routing
  - GnuGk ACD
- Billing
  - StatusAcct module
  - interface to external billing applications
- many proprietary / custom applications

Ways to route calls

- Gateway selection (config)
- Destination rewriting (config)
- Call Failover (config)
- Virtual queues (external)
- Radius based (external)
- use / chain the routing policies to configure which of the above are active

Virtual Queues

- no queued calls with announcements etc.
- „external ARQ rewriting“
- Config
  - define list or regexp of destinations to route
- Event
  - RouteRequest
- Commands
  - RouteReject (disconnect call)
  - RouteToAlias (change destination alias)
  - RouteToGateway (change destination alias and destination IP „out-of-zone routing“)

What's new in GnuGk 2.2.4 ?

- Call failover
- ENUM routing fully working
- CapacityControl module
- StatusAcct module (accounting via TCP/IP)
- RewriteCLI module (change or hide caller id)
  - started in 2.2.3

Call Failover

- when multiple routes are available and one routes failes, other route is used
  - round-robin or priorities for certain routes
- currently works for gateways
  - triggered by Q.931 cause codes
- can be extended to neighbors, VQs etc.
- high ASR even with cheap routes
- new in GnuGk 2.2.4

ENUM

- new routing policy
- use in chain of policies
- tested with Swiss ENUM Provider (Switch.ch)
- use GnuGk >= 2.2.4

RewriteCLI

- rewrite or hide caller id
  - inbound and outbound rules
  - based on IPs, number ranges etc.
- new in GnuGk 2.2.3

GnuGk Performance

- depends strongly on the gatekeeper mode (direct, routed signalling, full proxy)
- direct and routed modes are able to process a few thousands of simultaneous calls on a typical high-end PC machine
- full proxy mode is designed for small call volumes - a few hundreds of simultaneous calls
- for large volume of calls a Unix version of GnuGk is recommended

Performance Optimization (1)

- use LARGE_FDSET=... for large call volumes
  - only on Unix
  - compiletime config
  - stresses CPU less than OpenH323 socket handling
  - LARGE_FDSET=1024 for <= 100 concurrent calls
  - rule of thumb:
    - max. concurrent calls * 10 + 20%
    - 10 sockets/call: 2 for Q.931 + 2 for H.245 + 6 for RTP etc.
  - usually an OS limit for maximum number of file handles per process needs to be increased (using 'ulimit' command, for example) to match the new LARGE_FDSET value

Performance Optimization (2)

- GnuGk spawns one or more threads (signaling handlers) to handle signaling messages and perform authorization / accounting
- for best call throughput (and max. concurrent calls) tune CallSignalHandlerNumber / RtpHandlerNumber variable
  - runtime config
  - Windows PWLib has default limit of 64 sockets / thread
    - or recompile PWLib with FD_SETSIZE=x macro
  - don't let a single signaling handler to handle too many calls
    - CallSignalHandlerNumber=ConcurrentCalls/10

Performance Bottlenecks

- slow accounting/authorization backend (a database without indexes or with inefficient ones, queries not optimized, no RAID disks, RADIUS server runs out of resources, etc...)
- excessive network packet throughput:
  - a single G.723.1 call requires (at most) 70 UDP packets/s to be sent/received (in both directions) from each party:
    - 140 packets/s per call => 45.000 packets/s for 300 concurrent calls
    - add 5% for signaling => ca. 50.000 packet/s for 300 calls
  - Gigabit Ethernet cards can handle high packet rates without triggering too much interrupts to the kernel

GnuGk Future

- H.323 is not dead
- even more flexible call routing:
  - smart route selection (LCR – Least Cost Routing)
  - multistage E.164 number rewritting
- more advanced gatekeeper clustering
- dialler applications
- development of external applications on top of GnuGk

Visit http://www.gnugk.org

Thank you!

GnuGk

GNU Gatekeeper