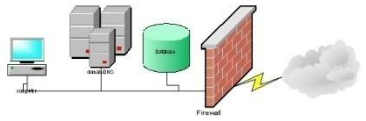
IT Services

General requirements

&

principles

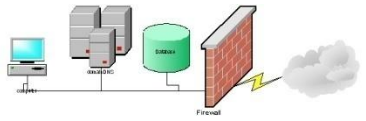
BME VIK TMIT



IT Services - overview

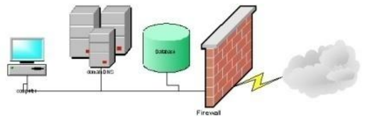
- In a company the IT task appears in more levels
- Example: *one* company, *one* IT group
- IT services
  - Planning
  - Install
  - Development
    - Reliability!
    - Scalability!
  - Monitoring
  - Maintenance
  - ...Support...

„If a >>service<< is not under monitoring,
Then that is not a Service.”



IT Services – overview ‘2

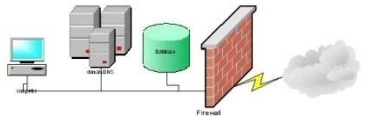
- These are all over the company’s area...
- Basic services (critical and visible)
  - Network connection
  - DNS
  - **Email**
  - Authentication
  - **Remote access**
  - **Printing**
- Additional services
  - Developer tools
  - Licence-handling
  - Shared storage
  - Shared calendar
  - Backup services, ...



Fundamental questions

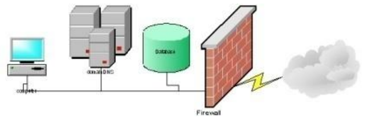
- Reliability
- User requirement
- Server quality machines, server room
- Servers’ basic features
  - simplicity
  - significant security
- Dependence of services
  - eg: business process - email - DNS – network
  - eg2.: from authentication service
- Accessibilty to servesr: only „SA”

If a fundamental service falls down, the dependent is affected!



Basic requirements' „fundamental"-handling

- The well maintened service should:
  - Be Simple
  - Contain less dependence
  - „Standard" HW
  - „Standard" SW
  - Standardized configurations
  - Documented, in a „standardized" place
  - Independent from the master machine's HW
    - Use ***service*** *names on the clients*!

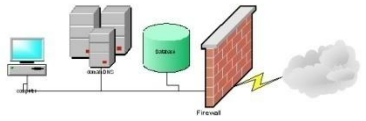


User requirements

- We build the services for them!
- Find out their expectations – work out it witrhin rational boarders.
- Define and propagate

  service level (SLA)

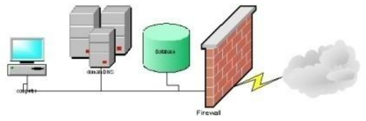
  – certain misunderstandings can be avoided
- Clarify with users, which *SLA* is limited why



Functioning requirement

- **Fat Client**
  - The application mainly runs on the users machine
- **Thin Client**
  - The main part of the application run on the server
- **Upgrade path**

- How much
  - They rely on the network?
  - They load the network?



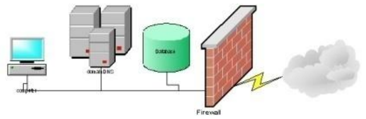
open - or – own components?

- If possible, use
  - Standard
  - Open
  - Tested

  components!
- *Protocol*
  - This is the communication form (possibly standardized)
  - „extended protocol” – different from standard
- *Protocol-realization*
  - Part of a product, different from standard
  - *two protocol-realization rarely „understand each other”*
- *Protocol-gateway*
  - *Sometimes required but better to keep off the system (SPF)*

Single Point of Failure

BME VIK TMIT

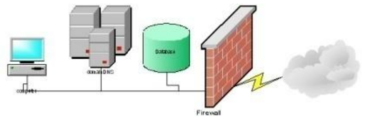


Simplicity

- Plan/build simple things
- Be limited to basic functions
  - Minimize user-configurable possibilities!
- It will be complex with the growth of the system

Reliability

- *Simplicity*...!
- Redundancy
  - Redundant HW; efficient utilization
    - Eg.: one machine with two power-supply – different power source
    - ...different sites!
- Non-redundant service-elements
  - Be clamped!
    - Limited functionality, less SPF
    - Equivalent power supply
    - Equivalent network elements/servers
- Smaller-bigger outage
  - Soft outage / hard outage



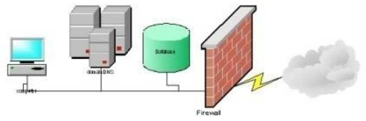
Centralization and Standardized-solutions

- Maintenance, monitoring, pricing points of view - it is worth considering :
  - Tools
  - Applications
  - Services

$\rightarrow$ **Centralization**

- Geographical or organization borders may not be so important - so step over

(more centralized points possible)



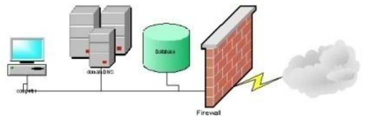
Performance

- „Working”... and fast enough?

  - Reliable
  - Functionally suit?
  - Pleasant GUI?
  - ...

- The expectations become more acute, when the network, graphic, processor,... accelerate
- Bad capacity plan – bad first *experience*
  - Server choosing:
    - Disk capacity?
    - Memory?
    - Processor?

$\longrightarrow$ Service requirements

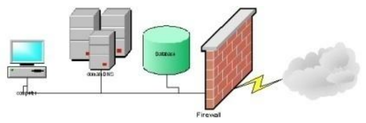


Service-monitoring

...Monitoring...!

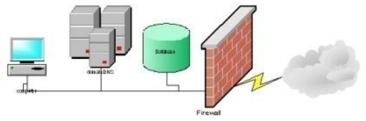
see eg.

- Fault Mgmt,
- Performance Mgmt

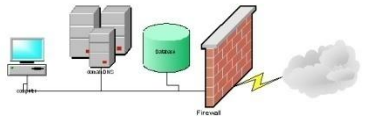


Over the...

- Extra special care -> extra results
  - Service installation
    - Complete documentation accessible
    - Trainig
    - The assistant and support staff prepared
  - Dedicated devices for each service
  - Complete redundancy
    - The whole service can take over another server (group)



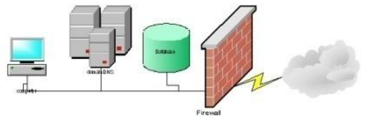
E-mail service



Did not get the e-mail?!

- Complete companies *rest* on secure e-mail services

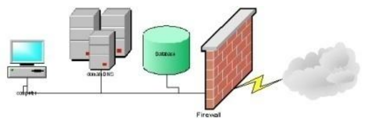
- Besides this...
  - Scalable
  - Simple, clearly perspicuous
  - General and uniform
  - Automated
  - Secure
  - Archived



The steps of sending e-mail

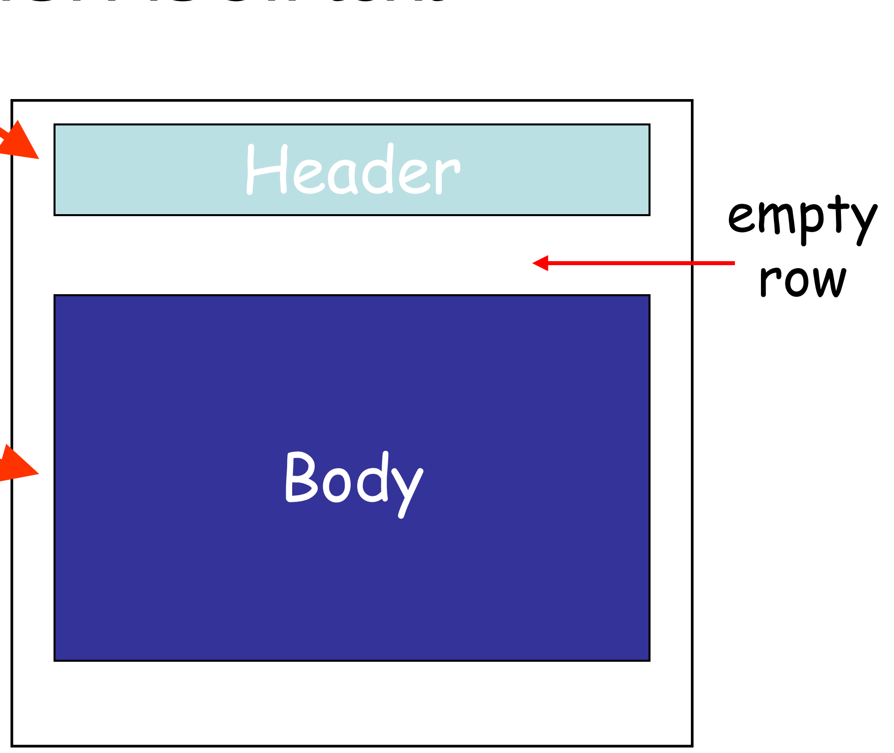
Differentation:

- Message forwarding
  - As e-mail is transferred from server to server
- Delivery
  - When e-mail is getting in to the receivers' mailbox
- Release message-lists
  - When the sent mail gets to the message-list, multiplied and forwarded



E-Mail message form (RFC 822)

- The message consists two parts
  - Header, Coding: 7-bit U.S. ASCII text
  - Body, Coding: 7-bit U.S. ASCII text
- Header
  - “type: value” shaped rows
  - “To: iru.bme@gmail.com”
  - “Subject: Meresek”
- Body
  - The text message
  - Non structured

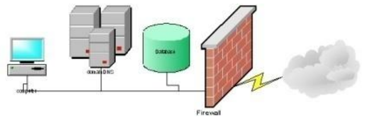


Restriction: Non-textual data sending...

- E-mail body 7-bit U.S. ASCII
  - If somebody wants to send non English text?
  - ...binary files (eg. Pictures, .exe-s) ?
- Solution: convert... non-ASCII -> ASCII
  - Base64 coding: every 3 byte form 4 printable U.S.-ASCII caracter
  - Uuencode (Unix-to-Unix Encoding) widespread

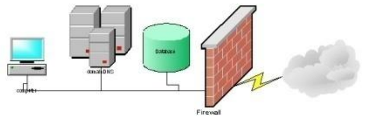
```
begin 644 cat.txt
#0V%T
`

end
```

  - Limit: the file-name the only „tip” to the data type...



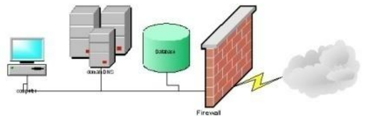
Restriction: Sending more data unit

- The users often wants to send more and different data in one message
  - More pictures, powerpoint file, or e-mail text message
  - The e-mail body single datafile
- Example: e-mail digest
  - We can pack more e-mail messages into one big message
  - Often used on big e-mail lists
- More solutions were born - to separate the parts
  - Eg. „well-known” separator-string between the parts
  - We need a standard method...



Multipurpose Internet Mail Extensions

- Added headers for the description of the body
  - MIME-Version: which MIME version is used
  - Content-Type: what kind of data type is in the body
  - Content-Transfer-Encoding: how to code the data?
- Content-types and sub-types definition
  - Eg. image – sub type: gif, jpeg
  - Eg. text – sub type: plain, html, and richtext
  - Eg. application – sub type: postscript and msword
  - Eg. multipart – message contains more data types
- How does it code the data to ASCII format?
  - Base64 coding: uuencode/uudecode



Example: E-Mail message MIME-reading

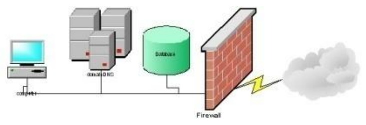
MIME version

Datacoding type

type and sub type

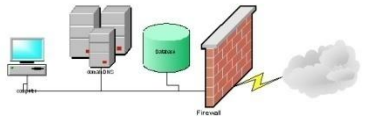
Coded data

```
From: jrex@cs.princeton.edu
To: feamster@cc.gatech.edu
Subject: picture of Thomas Sweet
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.........................
......base64 encoded data
```

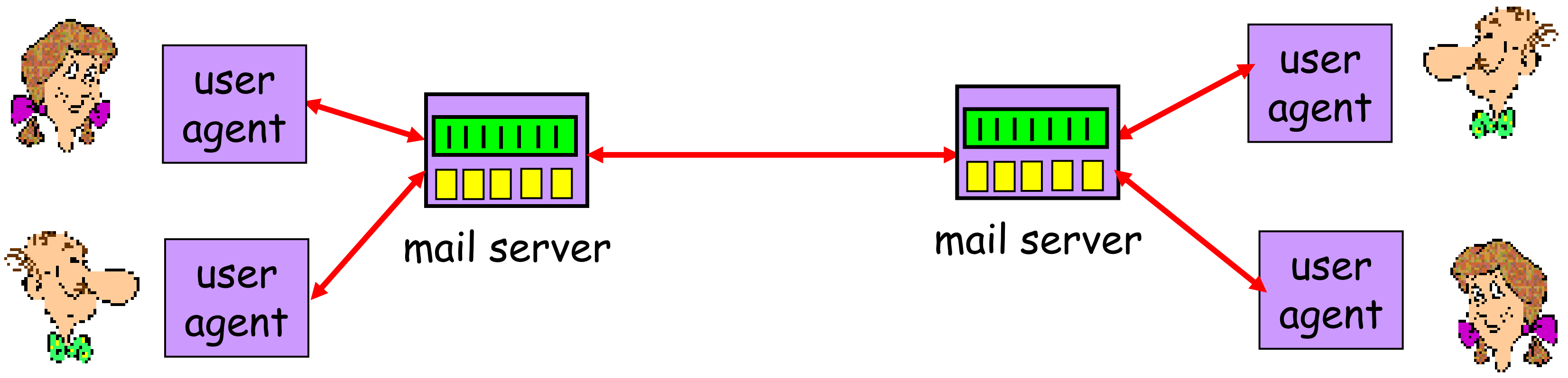


E-Mail addresses

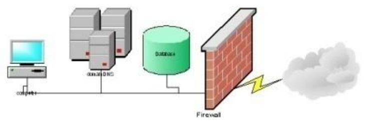
- The components of the e-mail address
  - Local mailbox (eg. pvarga or john.smith)
  - Domain name (eg. tmit.bme.hu)
- The domain name does not necessarily link to the mail server
  - The mail server has longer/encrypted name
    - Eg. tmit.bme.hu vs. mail.tmit.bme.hu
  - More servers can be covered (fault tolerance)
    - Eg. cnn.com vs. atlmail3.turner.com and nycmail2.turner.com
- Identification of the mail server connected to the domain
  - DNS request, by MX records (Mail eXchange)
    - Eg.: nslookup -q=mx tmit.bme.hu
  - Conventional DNS request for the IP address



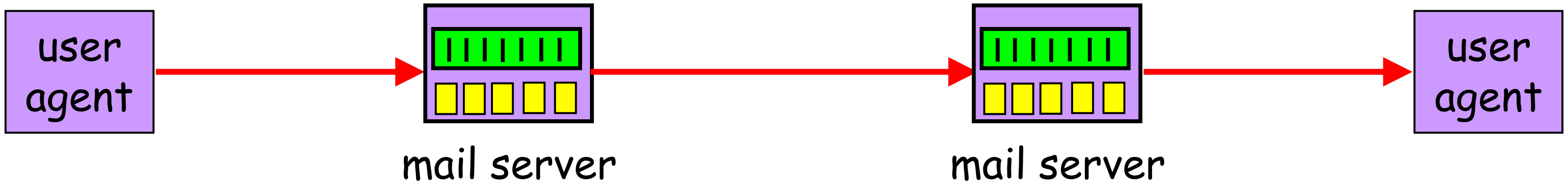
Mail Servers and User Agents



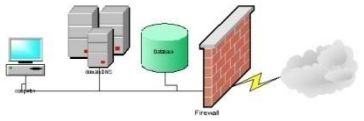
- Mail servers
  - Always online and accessible
  - e-mail „transportation” from other servers / to other servers
- User agents
  - Sometimes online and sometimes accessible
  - For users: different interfaces



SMTP
Store-and-Forward Protocol

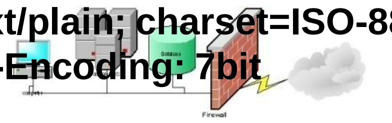


- The messages transferred by series of servers
  - The servers store the incoming messages in rows
  - … and when they get an opportunity, forward to the next hop
- If the next is unavailable
  - The server stores the messages; later tries again
- Every „hop” writes the identifier in to the message
  - The “Received” header helps to find the faults



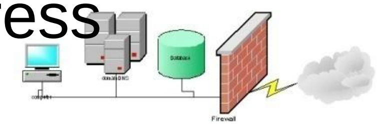
Example, **Received** Header

```
Return-Path: <casado@cs.stanford.edu>
Received: from ribavirin.CS.Princeton.EDU (ribavirin.CS.Princeton.EDU [128.112.136.44])
        by newark.CS.Princeton.EDU (8.12.11/8.12.11) with SMTP id k04M5R7Y023164
        for <jrex@newark.CS.Princeton.EDU>; Wed, 4 Jan 2006 17:05:37 -0500 (EST)
Received: from bluebox.CS.Princeton.EDU ([128.112.136.38])
        by ribavirin.CS.Princeton.EDU (SMSSMTP 4.1.0.19) with SMTP id M2006010417053607946
        for <jrex@newark.CS.Princeton.EDU>; Wed, 04 Jan 2006 17:05:36 -0500
Received: from smtp-roam.Stanford.EDU (smtp-roam.Stanford.EDU [171.64.10.152])
        by bluebox.CS.Princeton.EDU (8.12.11/8.12.11) with ESMTP id k04M5XNQ005204
        for <jrex@cs.princeton.edu>; Wed, 4 Jan 2006 17:05:35 -0500 (EST)
Received: from [192.168.1.101] (adsl-69-107-78-147.dsl.pltn13.pacbell.net [69.107.78.147])
        (authenticated bits=0)
        by smtp-roam.Stanford.EDU (8.12.11/8.12.11) with ESMTP id k04M5W92018875
        (version=TLSv1/SSLv3 cipher=DHE-RSA-AES256-SHA bits=256 verify=NOT);
        Wed, 4 Jan 2006 14:05:32 -0800
Message-ID: <43BC46AF.3030306@cs.stanford.edu>
Date: Wed, 04 Jan 2006 14:05:35 -0800
From: Martin Casado <casado@cs.stanford.edu>
User-Agent: Mozilla Thunderbird 1.0 (Windows/20041206)
MIME-Version: 1.0
To: jrex@CS.Princeton.EDU
CC: Martin Casado <casado@cs.stanford.edu>
Subject: Using VNS in Class
Content-Type: text/plain; charset=ISO-8859-1; format=flowed
Content-Transfer-Encoding: 7bit
```



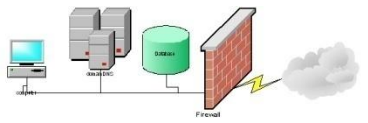
Multiple Server-hops

- Typically minimum two mail servers
  - Sender and receiver-side
- More
  - Separate servers for key functions
    - Spam filtering
    - Virus scan
  - Message-forwarding servers
    - pvarga@tmit.bme.hu -> pvarga@alpha.tmit.bme.hu
  - Electronical message („mailing”) lists
    - The message forwards to the list server
    - … from there the list come unbound -> to all list address

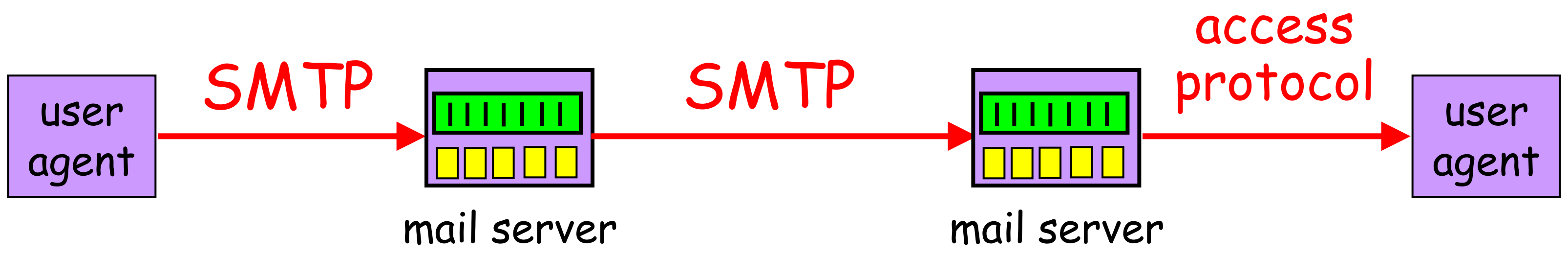


Electronical message-lists

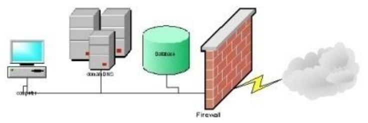
- User groups can be reached under 1 address
- Messages flood
  - From one e-mail there will be a lot of sent messages
  - one message copy as a receiver
- Handling rebound messages
  - The rebound can be from numerous reasons
  - Eg. receiver mailbox does not exist; limited resources,...
- E-mail digests
  - Sending in one the mails of message list
  - Between messages has separator character rows
  - … or sended in a multiple/digest form



Simple Mail Transfer Protocol

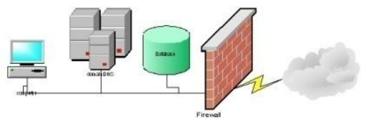


- Client-server protocol
  - *Client:* the sender mail server
  - *Server:* the receiver mail server
- Secure data transmission
  - Over TCP (on port 25)
- „Push” protocol
  - The sender server pushes the file into the receiver server
  - … insted of waiting for the receivers request



Simple Mail Transfer Protocol

- Command/response
  - Command: ASCII text
  - Response: 3-digit state-code and text
- Synch
  - The sender is waiting for a response to the „command”
  - … before sending the new „command”
- The 3 phases of transferring
  - Handshaking
  - Message tarnsfer
  - Termination



Example:
Ann sends a messeage to Béla

1) Ann uses the UA, to write a message to here:
beela@nagyceg.com

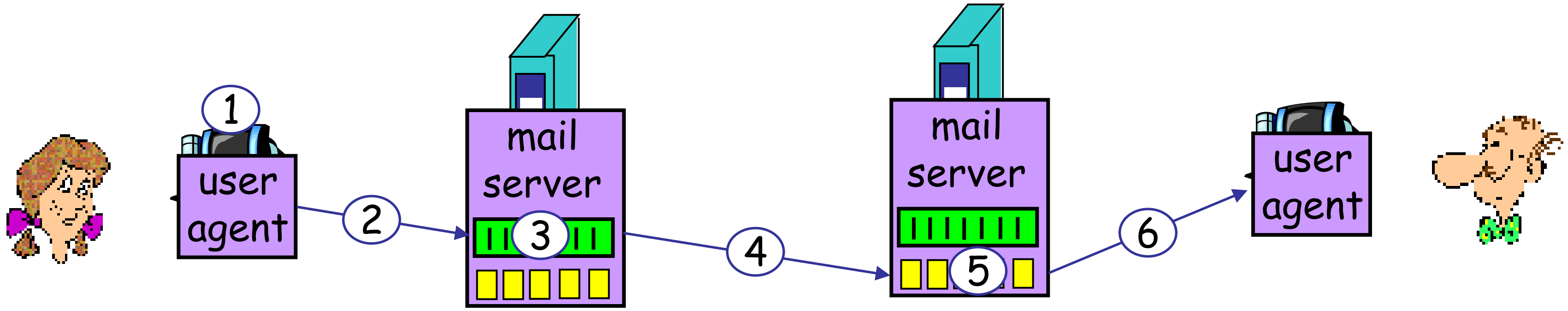
2) Anns' UA sends the message to the mail server

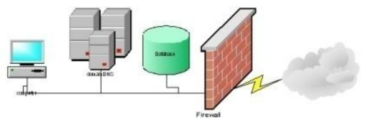
3) The client side's SMTP opens a TCP connection with Béla's mail server

4) The SMTP client sends Ann's message in the TCP connection

5) Béla's mail server puts the message into Béla's mailbox

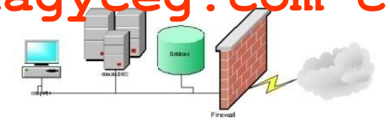
6) Béla activates his UA to read the message





Example: SMTP message transfer

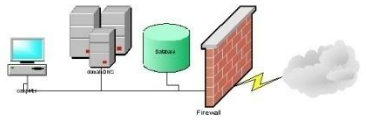
```
S: 220 nagyceg.com
C: HELO segitokez.hu
S: 250  Hello segitokez.hu, pleased to meet you
C: MAIL FROM: <anna@segitokez.hu>
S: 250 anna@segitokez.hu... Sender ok
C: RCPT TO: <beela@nagyceg.com>
S: 250 beela@nagyceg.com ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Dear Beela,
C: Thank you for the flowers!
C: Anna
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 nagyceg.com closing connection
```



Retrieve E-mail from the Server

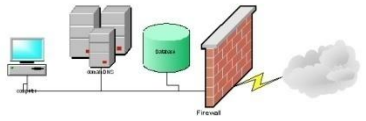
- The server stores the incoming mails in mailboxes
  - Selected by the filed “To”
- The user has to retrieve the e-mail
  - Asynchronous according to sender time
  - ...check it and answer
  - ...order and archive messages

- In the old times…
  - The user checks the PC where the mail arrived
  - The users read the mail in working places



The effect of PC for the E-Mail retrive methods

- Separate PC for personal usage
  - The user do not want to log in remote PCs
- Resource-limitation
  - Most PCs have not got enough resources to work as an e-mail server
- Periodic connection/accessibility
  - The PC-s rarely connected to the network
  - ...Because of the characteristics of Internet-connection's, and PC's power on/off
  - The server has to try to connect redundantly
- And ….: Post Office Protocol (POP)

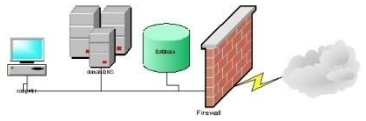


Post Office Protocol (POP)

- POP targets
  - Users connected with high frequency
  - User can retrieve their e-mails when connected
  - … and look them/manipulate them, when not connected (off-line)

- Typical user-agent interaction with the POP server
  - Connection to the server
  - Retrieve all e-mails
  - Store the messages as „new” in the PC
    - Delete messages from the server
  - Terminate connection with server

- The UA uses SMTP for the message sending



POP3 Protocol

Authorization phase

- Client „command”:
  - **`user:`** username
  - **`pass:`** password
- Server „response”
  - **`+OK`**
  - **`-ERR`**

```
S: +OK POP3 server ready
C: user beela
S: +OK
C: pass nagyfonokvagyok
S: +OK user successfully logged on
```

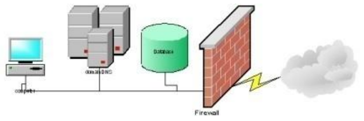
Transaction phase,

Client:

- **`list:`** list messages by number
- **`retr:`** retrieve by the number
- **`dele:`** delete
- **`quit`**

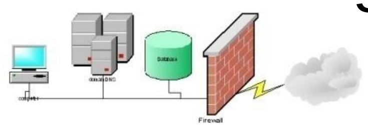
```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

BME VIK TMIT



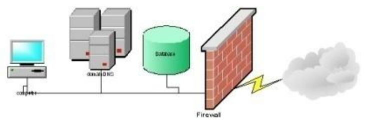
Limitations of POP

- Not easy to handle multiple mailboxes
  - It is designed to have the users' incoming e-mails in one place
- Not designed for storing messages on the server
  - ...but to download the messages to the client
- Hard to access with multiple clients to the mailbox
  - It is important, because the user has a home, workplace PC, laptop, cyber caffee PC, etc.
- Requires high network bandwith
  - Transfers all messages, often before reading



Interactive Mail Access Protocol (IMAP)

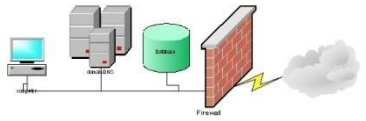
- „Connected” and „Disconnected” methods supported
  - The users can download the messages as they wish
- Simultaneously more client can connect to the mailbox
  - Detect the modifications by other clients on the mailbox
  - Server monitors and stores the state of messages (eg. unreaded, read, sent)
- Access to the MIME part of messages & partial download
  - The clients can download the MIME parts partially
  - Eg. The text part of the message – without the download of the attachment



Interactive Mail Access Protocol (IMAP)

- Multiple mailboxes on the server
  - The client can create, rename, and delete mailbox
  - The client can transfer messages from one folder to an other

- Server-side search
  - Before downloading the message, a search can start on the server



E-Mail by web

- User agent: conventional Web browser
  - The user communicates with server on HTTP
  - Eg.: Gmail, Yahoo mail, Hotmail, freemail,...
- E-mail reading
  - The Webpages display the content of folders
  - … and allow to see and download the messages
  - “GET” claims to display different Webpages
- E-mail sending
  - We write the text into a „form”, than „submit” to the server
  - “POST” request and data load up to server
  - The server sends message with SMTP to other server
- Easy to send an anonymous e-mail (Eg. spam)