JÜLICH
FORSCHUNGSZENTRUM

OpenGridForum
OPEN FORUM | OPEN STANDARDS

# Firewalls and Grids
# Update on UDP Hole Punching

R.Niederberger@fz-juelich.de
Th.Oistrez@fz-juelich.de

OGF 23 - FI-RG - 05.06.2008

# Overview

- A Firewall and Grids overview
- The principle design of FUHP
- An experimental application
- Summary

# Grid requirements

- A Grid is a union of geographically distributed, independent organizations
- Dynamic use of resources, often in parallel

**The initial problem:**

- Internal hosts are protected by local firewalls
- Often only outgoing connections are allowed
- Having a client and server model implies one of both has to have an incoming connection
- So none can start communication

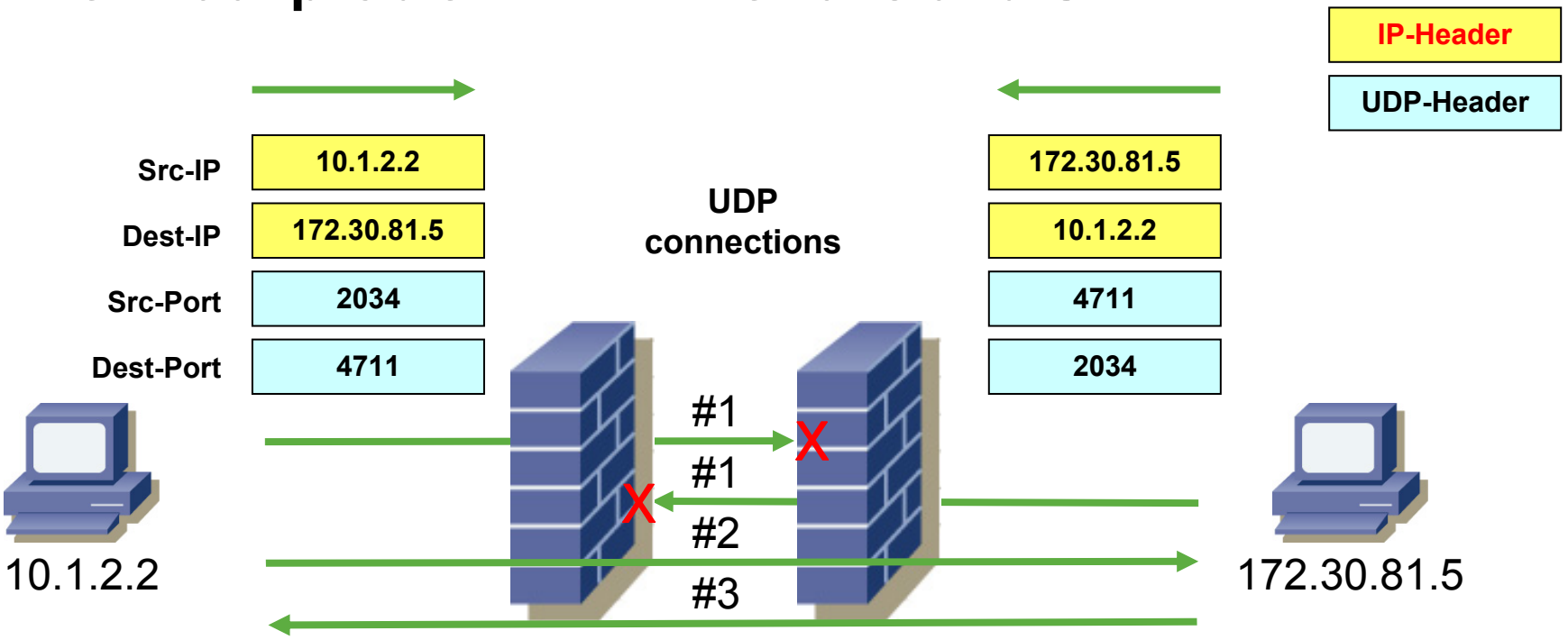
# Solution requirements

- Integration in existing security concept
- Usable in open source and commercial environments
- Communication between partners only for minimum necessary duration

# The principle design of Firewall UDP Hole Punching (FUHP)

# Filtering of traffic

## The initial problem: Firewalls and UDP




**Neither Client nor Server can reach the other one (#1)**

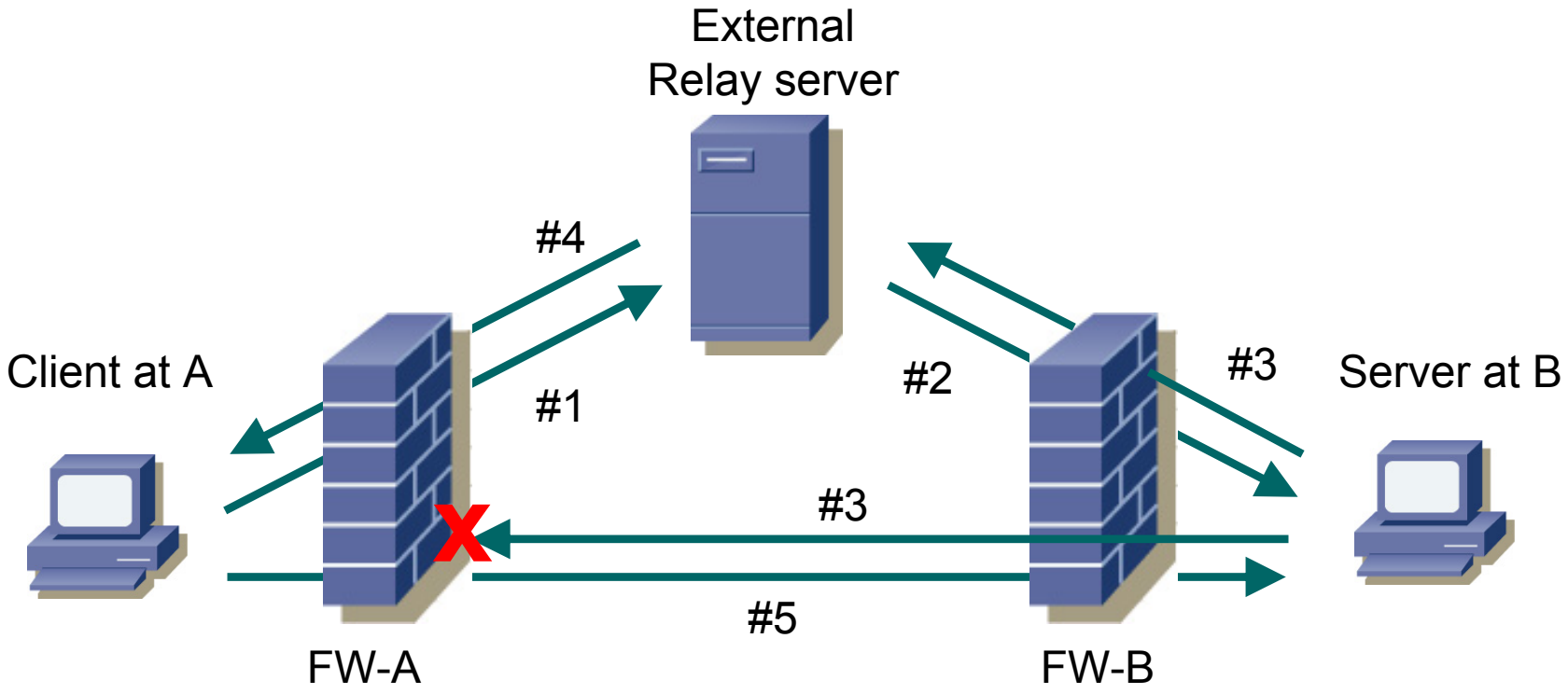
**After one of both has initiated and the other knows about this, he can answer (#1, #2, #3)**

# first solution

- put server outside both firewalls
- harden OS system and allow only specific communication ports
- this server has crtl connections to client and server
- after having checked authenticity and authorization,
  outside server tells inside server about connection request
  from client (including client-ip and client-port info)
- inside server initiates connection to client using client-ip and -port info
  -> firewall at server side allows outgoing connection
  -> firewall at client side rejects connection
- additionally, client now connects to server, but gets through firewall at server side (server already opened this hole), because firewall at server side assumes packets from client to be answers to connection initiated by server

# Dynamic configuration of Firewalls

## The UDP hole punching concept



# Simple solution, works quite well, but …

external relay server needed

- bastion host
- who administrates this server (OS and security)?
- for every service/every installation one server?

- outgoing connections have to be allowed
- works only with UDP (TCP sequence number problem)
- where is checked what and who is allowed (external or internal)?
- relay server has to handle double traffic rate per connection
- relay server has to handle multiple connections in parallel
- tables of known services have to be managed at outside server
- generalization ? (ip addr. of external servers have to be well known)

# Simple improvements, now it‘s working better …

## Solutions

- Combine external server and internal service at one internal host
- open well known port, e.g. TCP 4711 to access relay server
- encrypted communication between client and relay server
- internal communication between relay server and service
- check service dependent internally: authentication & authorization
- outgoing connections have to be allowed (<u>further on required</u>)
- works only with UDP → this implies:
  - use UDP datagramms for messages
  - use UDT (UDP-based Data Transfer Protocol) for files

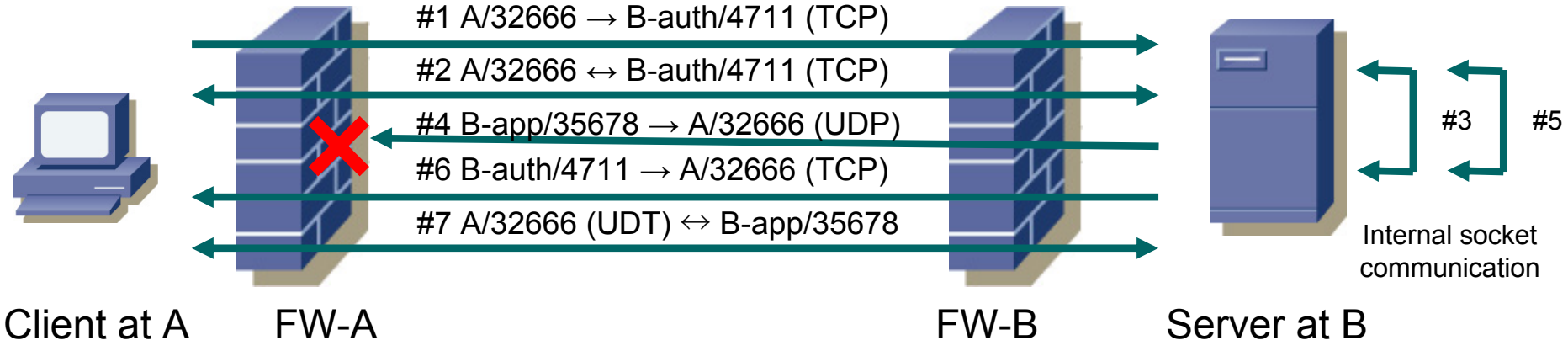
# Dynamic configuration of Firewalls

## The UDP hole punching concept

## in Grid environments

#3 exchange infos between auth server and app server

#5 tell auth server about used port and already started communication



# http://udt.sourceforge.net/

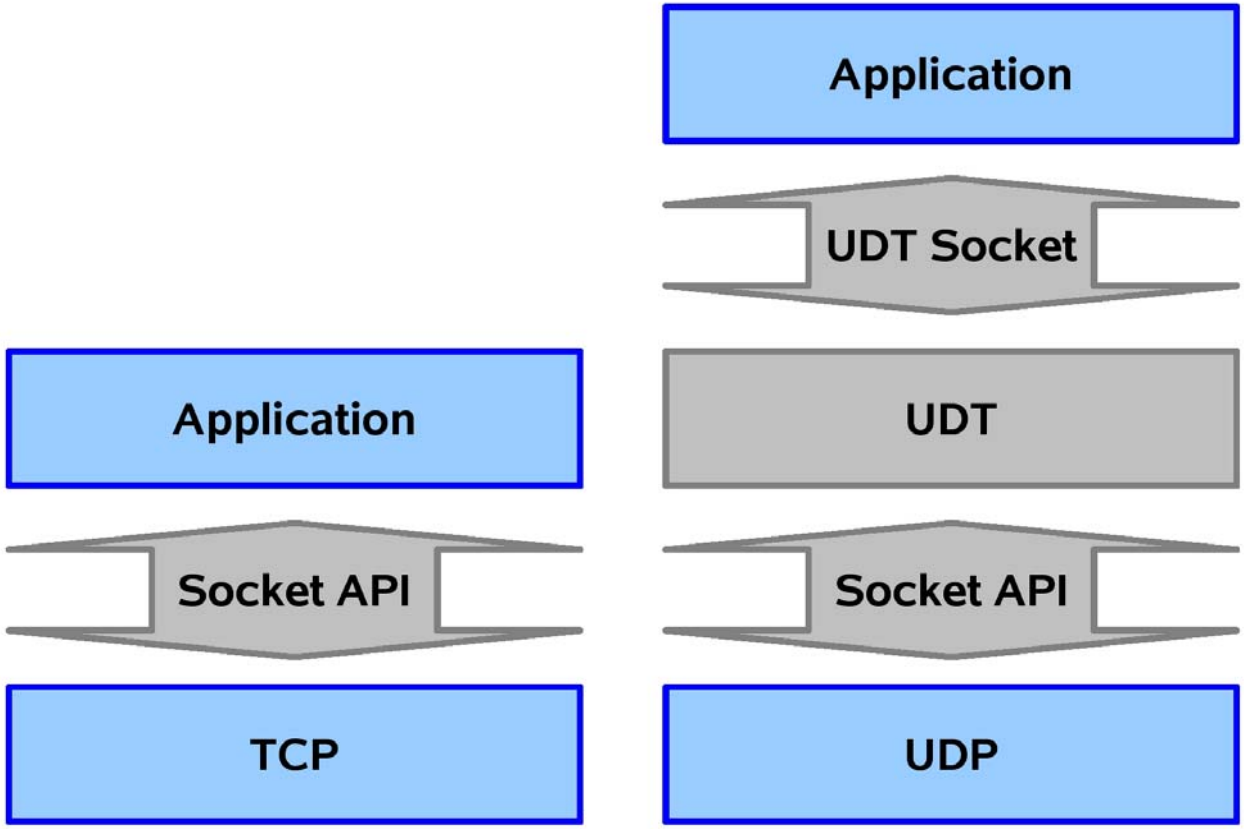
**UDT: Breaking the Data Transfer Bottleneck**

UDT is an application level data transport protocol for emerging distributed data intensive applications over wide area high-speed networks. UDT uses UDP to transfer bulk data and it has its own reliability control and congestion control mechanisms.

**Key Features**

- **Fast**. UDT is designed for extremely high speed networks and it has been used to support global data transfer of terabyte sized data sets.
- **Fair and Friendly**. Concurrent UDT flows can share the available bandwidth fairly, while UDT also leaves enough bandwidth for TCP.
- **Easy to Use**. UDT resides completely at the application level. Users can simply download the software and start to use it. No kernel reconfiguration is needed. In addition, UDT's API is very similar to the traditional socket API so that existing applications can be easily modified.
- **Highly Configurable**. UDT supports user defined congestion control algorithms with a simple configuration. Users may also modify UDT to suit various situations. This feature can also be used by students and researchers to investigate new control algorithms.
- **Firewall Friendly**. UDT is completely based on UDP, which makes it easier to traverse the firewall. Starting from UDTv4, multiple UDT flows can share one UDP port, thus a firewall can open only one UDP port for all UDT connections. UDT also supports rendezvous connection setup.

# UDT implementation



# A proof of concept implementation

In the meantime within a Bachelor thesis an implementation has been done using the above concept

a.) A simple client/server application has been designed allowing a file transfer using this technique

b) An integration into a UNICORE environment has been implemented

The fact that UNICORE and its WebServices are implemented in JAVA, but UDT is implemented in C++ has lead to a hybrid solution using the Java Native Interface (JNI). JAVA code uses JNI to load dynamic native libraries and to involve their native functions. Therefore Wrapper functions had been necessary to built a bridge between the JAVA classes and the UDT interface.

The outcome of these implementations have been very encouraging showing that this concept works quite well and provides a fast alternative to transfer huge among of data between grid applications via dynamic port ranges, i.e. a good alternative to gridftp

# Status of proof of concept implementation

Dynamic opening of Firewall ports works fine.

File transfer provides high throughput (more than gridFTP), if configured correctly.

Security, i.e. authentication and authorization, has not been implemented yet. **!!!**

# Summary

Grid applications need high performance and low latencies.

Often multiple parallel connections are used to speed up the transfer.

To match this requirement of Grid applications static port ranges are opened on firewalls.

Dynamic configuration would ease this problem.

FUHP introduced a possible solution which configures a firewall dynamically based on UDP hole punching.

The concept has been adapted to the needs of Grid environments and the design of an implementation for UNICORE has been outlined.

The described concept of Grid UDP hole punching can be seen as an interim solution providing dynamic configuration of Firewalls for Grid applications until “real” dynamic configurable firewalls are available on the market.

# Questions and discussion