International *Mathematica*
User Conference 2008

Remote Kernel Strategies

Sascha Kratky
kratky@unisoftwareplus.com
uni software plus GmbH
Austria

Outline of the Talk

- Motivation for using remote *Mathematica* kernels
- Remote *Mathematica* kernel launch procedure
- Scenarios and strategies for using remote kernels
- Demo

Motivation for Using Remote Kernels

- *Mathematica* package platform availability
- Kernel performance (64-bit computing)
- *Mathematica* package development and testing
- grid*Mathematica*

Problems Connecting To Remote Kernel

Statements from *MathGroup*:

"There is a definite lack of error messages and trouble shooting documentation for incorrectly configured remote kernels."

"Connecting to a Unix kernel on a remote host under OS X is much more complicated than it should be."

Problems Connecting To Remote Kernel

- Lack of up-to-date documentation
- Configuration overhead
- Requires third party tools
- *Mathematica* front end often just stalls
- Working setup breaks easily

MathLink

- Peer-to-Peer based protocol
- TCPIP protocol used for remote communication
- lnk = LinkCreate[LinkProtocol->"TCPIP"] creates listening link *3563@192.168.1.15,3567@192.168.1.15*
- LinkConnect[lnk] connects to listening link
- each TCPIP MathLink uses two underlying TCP/IP connections

Kernel Configuration

Kernel name: New Kernel

☐ Append name to In/Out prompts

☐ Automatically launch on front end startup

◯ Basic Options

Launch on: ◯ Local machine ◉ Remote machine

Kernel program: math

Remote user: kratky

Remote host: hostname

◉ Advanced Options

Arguments to MLOpen:

```
-LinkMode Listen -LinkProtocol TCPIP -LinkOptions MLDontInteract
```

Launch command:

```
`java` -jar "`mathssh`" kratky@hostname math -mathlink -LinkMode
Connect -LinkProtocol TCPIP -LinkName "`linkname`" -LinkHost
`ipaddress`
```

☑ Translate Return into Newline

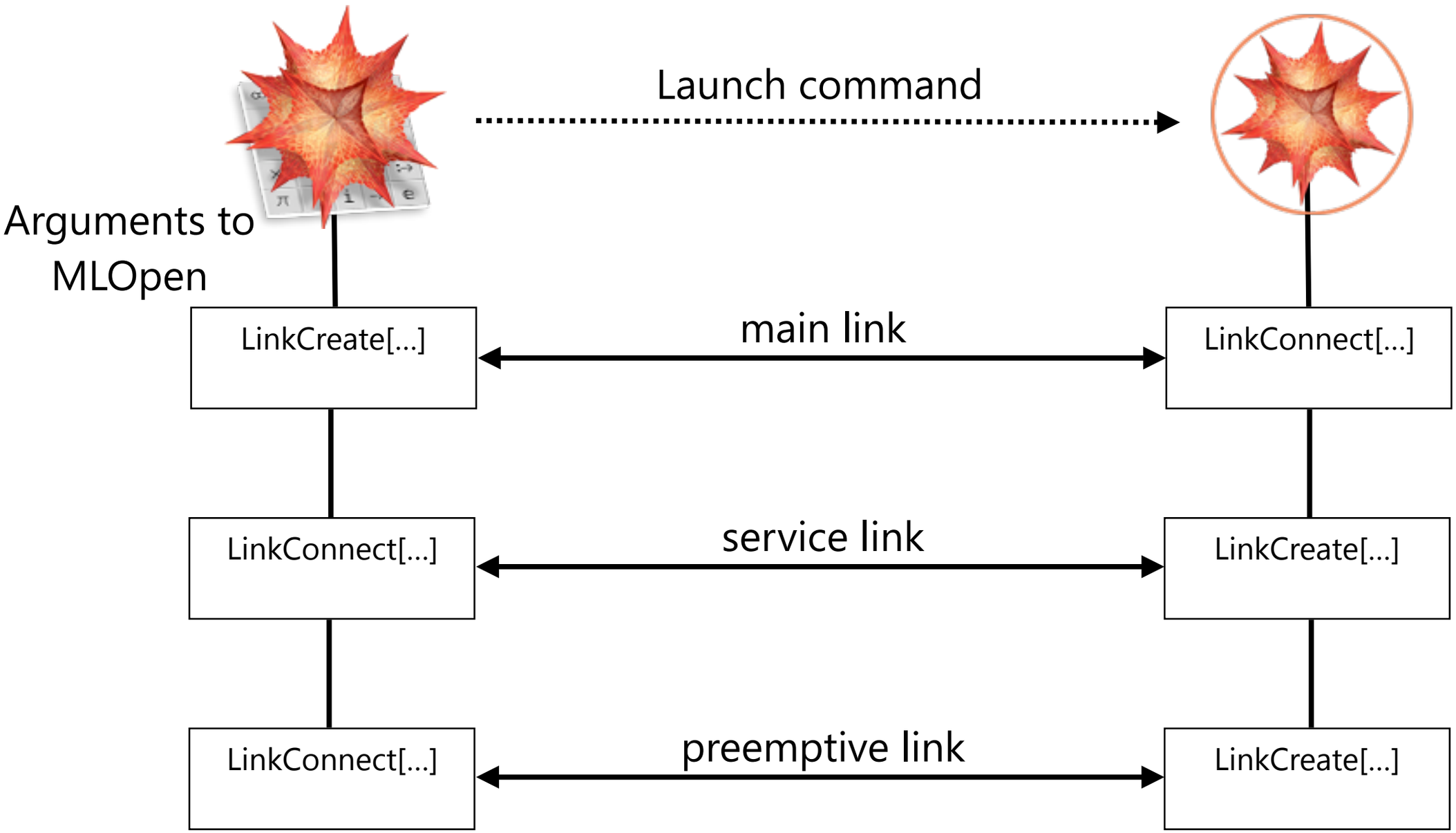
☐ Raw MathLink connection

?

Cancel OK

Local Machine

Remote Machine



Possible Points of Failure

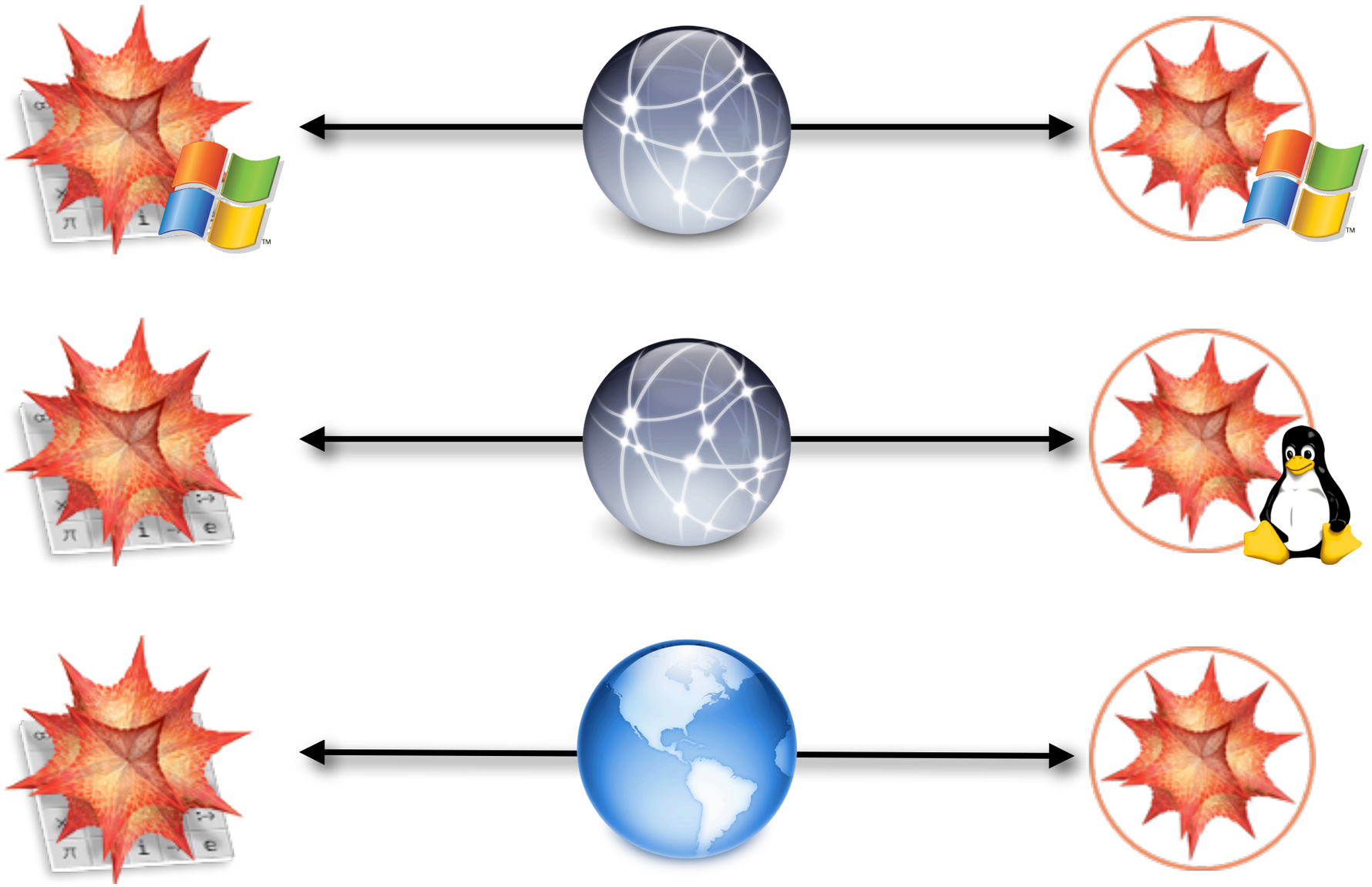
- Remote launch of kernel process fails
- Firewall on local machine prevents kernel callback from remote machine
- Firewall on remote machine prevents connection from local front end
- Arbitrary port numbers
- No route (multiple IP addresses)

Intranet

Internet

Front end and kernel machine on same local network	Front end and kernel machine on different local networks
No firewalls between front end and kernel machine	Firewalls between front end and kernel machine
Private IP address	Public IP ≠ Private IP (NAT)
Fast connection	Slow connection

Top 3 Scenarios

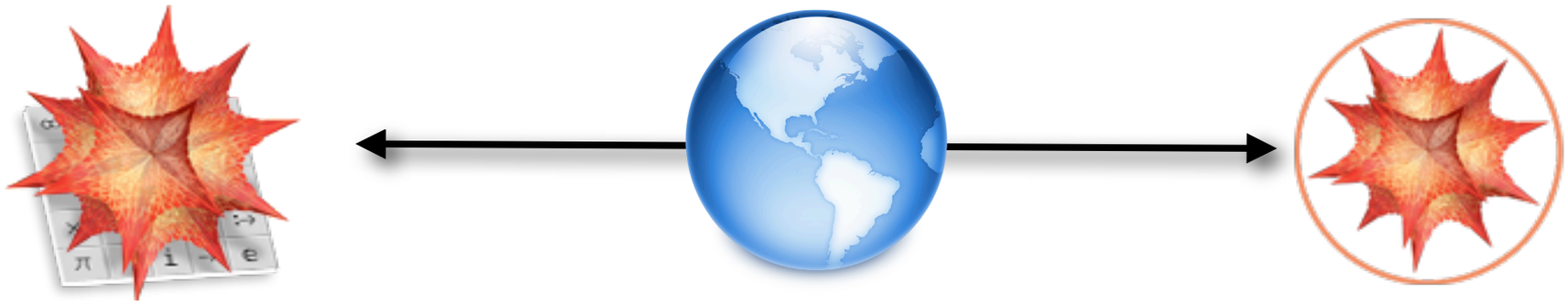


- Use PsExec to start remote kernel
- Free tool from Microsoft / Sysinternals
- No installation on remote Windows kernel machine required
- Talks to IPC$ share (NetBIOS)
- Supports Windows authentication

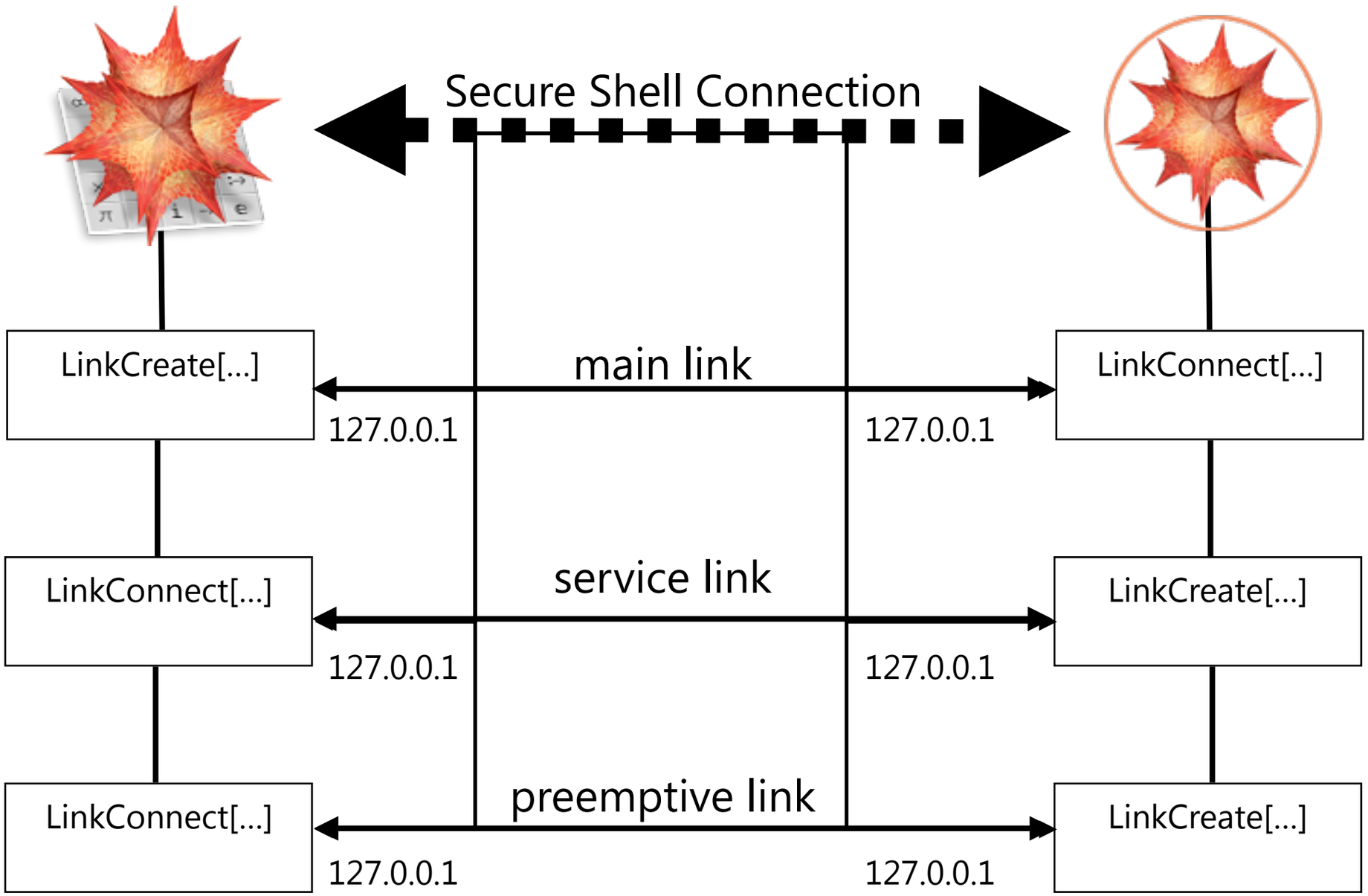
PsExec Configuration

- Check firewall settings
- Arguments to MLOpen:
  -LinkMode Listen -LinkProtocol TCPIP
  -LinkHost IP_address_pingable_from_remote_machine
  -LinkOptions MLDontInteract
- Launch command:
  psexec_installdir\psexec.exe -accepteula *remote_machine*
  -u *username* -p *password* "*path_to_mathematica_kernel*"
  -LinkMode Connect -LinkProtocol TCPIP
  -LinkName "`linkname`"

- Use Secure Shell to start remote kernel
- *Mathematica* ships with MathSSH
- SSH server needs to be running on remote Linux machine
- Password-less login via public key authentication
- Use WolframSSHKeyGen.jar to generate public/private key pair



- Solution should work for all platforms
- Must support NAT and activated firewalls
- Kernel launching should not require manual intervention
- Connection should be secure
- MathLink protocol data should be compressed


Local Machine
Remote Machine
Secure Shell Connection
LinkCreate[...]
LinkConnect[...]
main link
127.0.0.1
127.0.0.1
LinkConnect[...]
LinkCreate[...]
service link
127.0.0.1
127.0.0.1
LinkConnect[...]
LinkCreate[...]
preemptive link
127.0.0.1
127.0.0.1

Secure Shell Tunnel

- Kernel init file installed on remote kernel machine overrides service and preemptive link creation
- Linux / Mac OS X front end
  Shell script *tunnel.sh* launches kernel and sets up local and remote forwarding tunnels for links
- Windows front end
  Batch script *tunnel.bat* launches kernel uses third party SSH client PuTTy
- Creates log file *tunnel.log*

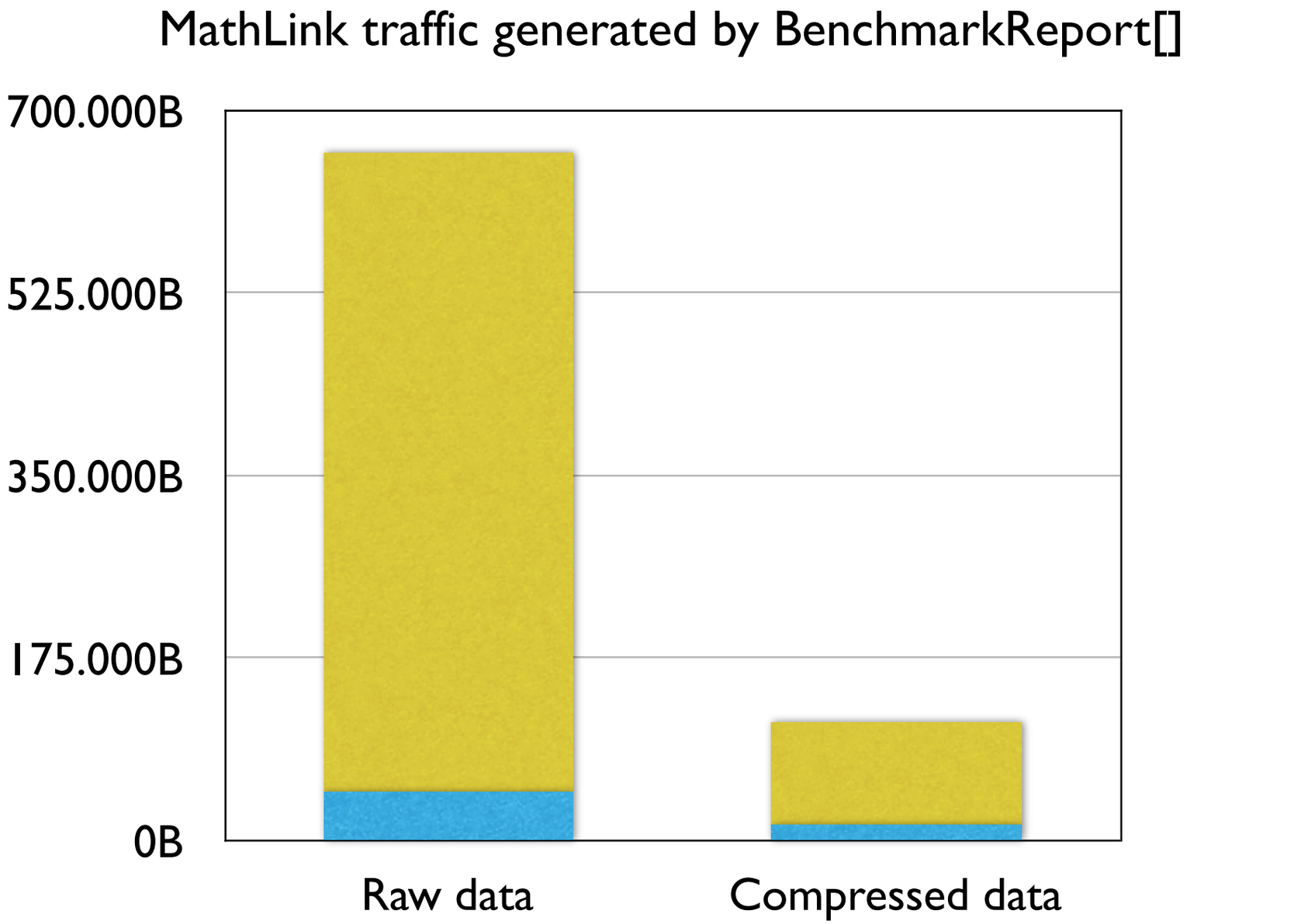
SSH Tunnel Configuration

- Arguments to MLOpen:
  -LinkMode Listen -LinkProtocol TCPIP
  -LinkHost 127.0.0.1
  -LinkOptions MLDontInteract

- Launch command (Windows):
  "`userbaseDirectory`\FrontEnd\tunnel.bat"
  [*user*[:*password*]@]*remote_machine*[:*port*] "*remote_kernel_path*"
  "`linkname`"

- Launch command (Mac OS X, Linux):
  "`userbaseDirectory`/FrontEnd/tunnel.sh"
  [*user*[:*password*]@]*remote_machine*[:*port*] "*remote_kernel_path*"
  "`linkname`"

SSH Compression



Demo