Distack

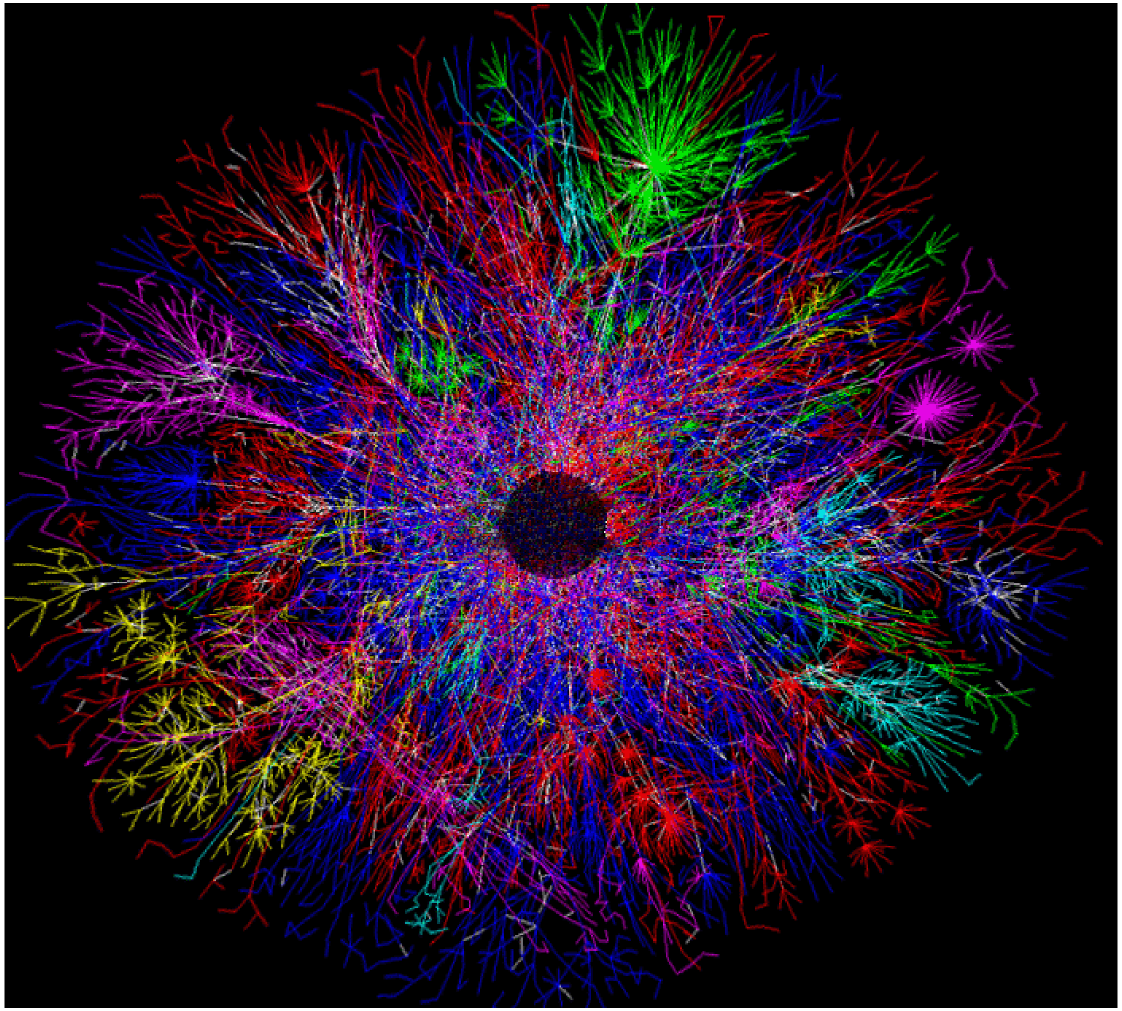
A Framework for Anomaly-based Large-scale Attack Detection

Thomas Gamer, Christoph P. Mayer, Martina Zitterbart

SECURWARE 2008, Cap Esterel, France

Institute of Telematics, University of Karlsruhe (TH)
Karlsruhe Institute of Technology (KIT)

Distributed Denial of Service

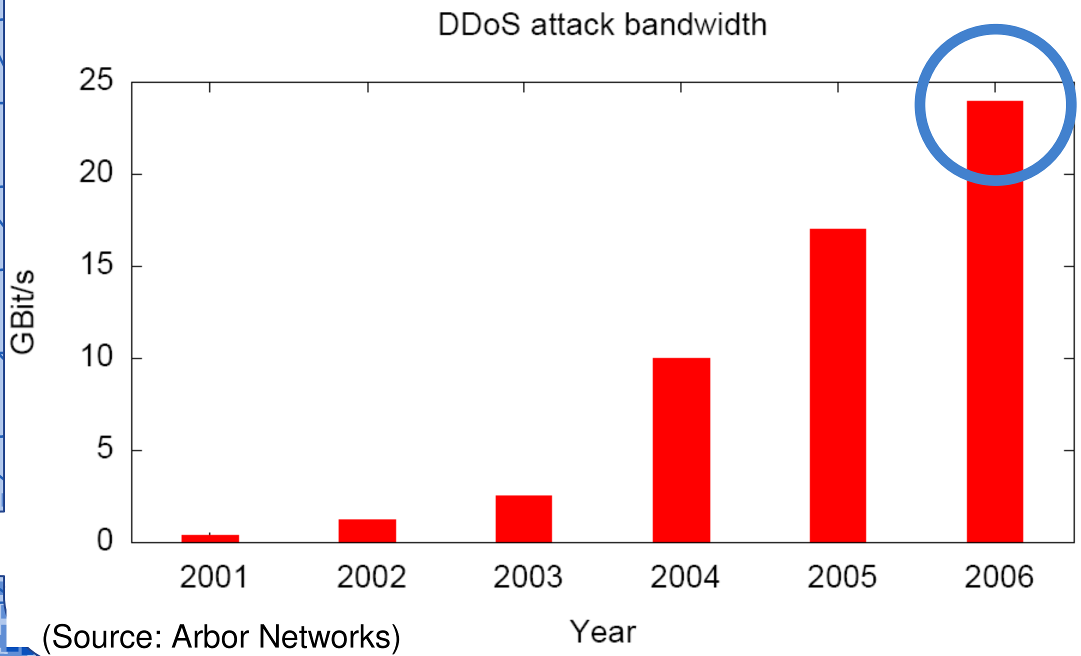


Source: Prolexic

DDoS – Huge threat to the Internet

„New Zealand teenager controlled botnet of ~~1.3~~ 50 million computers“ (Heise-Online, Nov. 2007)

„DDoS attacks and worms pose biggest threat to the Internet“ (Worldwide Infrastructure Security Report, Arbor Networks, 2007)



(Source: Arbor Networks)

1.3 million systems send at Ø 19kbit/s each

How can you detect and block such low traffic early?

→ Cooperation between detection instances seems promising!

Why can`t we cope with DDoS?

- Some exemplary issues
  - Little knowledge about global behavior of DDoS
  - Attacks highly distributed. Attack detection and countermeasures mostly not!
  - Few *directly* reusable results

Initial challenge:

Complex development and evaluation of mechanisms for local and *distributed* attack detection and traffic analysis

→ Initial development effort as base for your mechanisms is incredibly high!

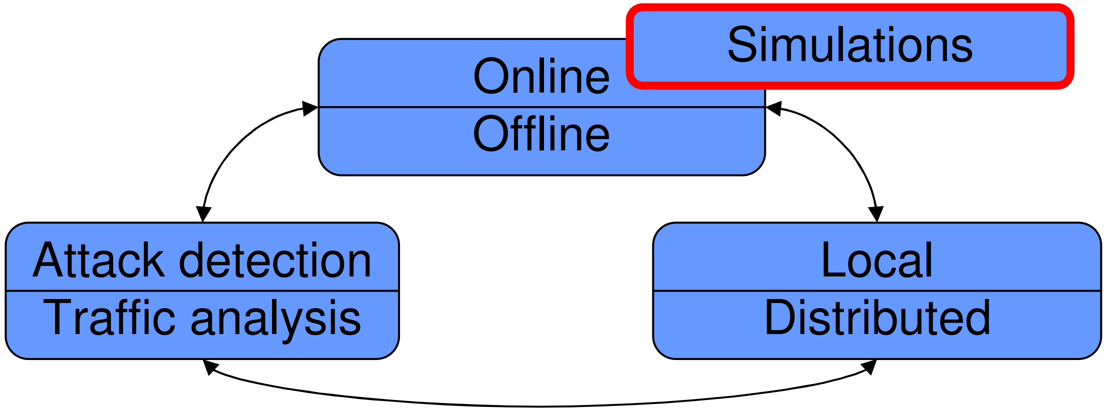
What you can do with Distack

- Attack detection and traffic analysis
  - Rapidly implement and run your attack detection and traffic analysis schemes
  - Lots of reusable modules (e.g. sampling, plotting)
  - Run on live traffic or captured traces
  - Comfortable communication between remote instances → easier distributed detection
- Simulations
  - Run your modules transparently in large-scale simulations
  - Integrates seamlessly with the toolkit OMNeT++/INET/ReaSE

and that`s not even all …

… even more fun with Distack

- Distack use-cases



- Examples
  - *Local traffic analysis*: easily analyze online traffic and traffic traces
  - *Distributed traffic analysis*: several measurement points in the network, report to a central instance

→ There is more than distributed attack detection!

Distack: Distributed attack detection

distack
distributed attack detection

Framework for distributed attack detection and traffic analysis

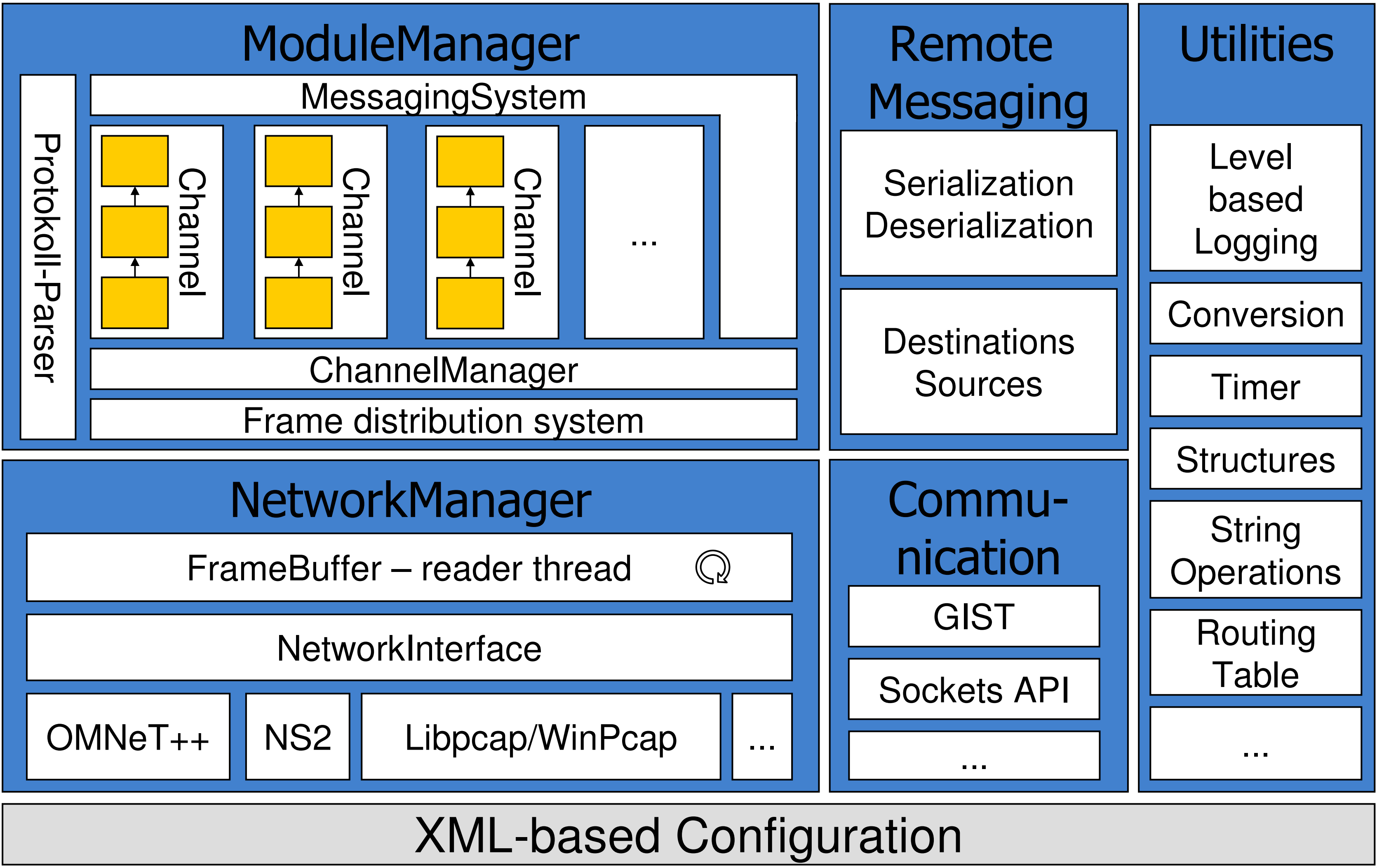
What it gives to *you*

- Fully concentrate on your methods for attack detection and traffic analysis
- Write once run everywhere: Transparently run your methods, e.g. on a PC or in a simulation environment
- High reuse through building blocks
- Great support for your attack detection

Rough Architectural Overview

- Module manager
  - Mechanisms are implemented in small building blocks → *modules*
  - The environment to implement your modules
- Network manager
  - Abstraction from the network
  - Handles the different ways packets come in
- Local and remote messaging
  - Communication for the lightweight modules
  - Data-centric communication, local and remote
- Configuration
  - Flexible way to configure your modules and Distack

Distack High-level Architecture



ModuleManager

Protokoll-Parser

MessagingSystem

Channel

Channel

Channel

...

ChannelManager

Frame distribution system

Remote Messaging

Serialization Deserialization

Destinations Sources

Utilities

Level based Logging

Conversion

Timer

Structures

String Operations

Routing Table

...

NetworkManager

FrameBuffer – reader thread

NetworkInterface

OMNeT++

NS2

Libpcap/WinPcap

...

Commu-nication

GIST

Sockets API

...

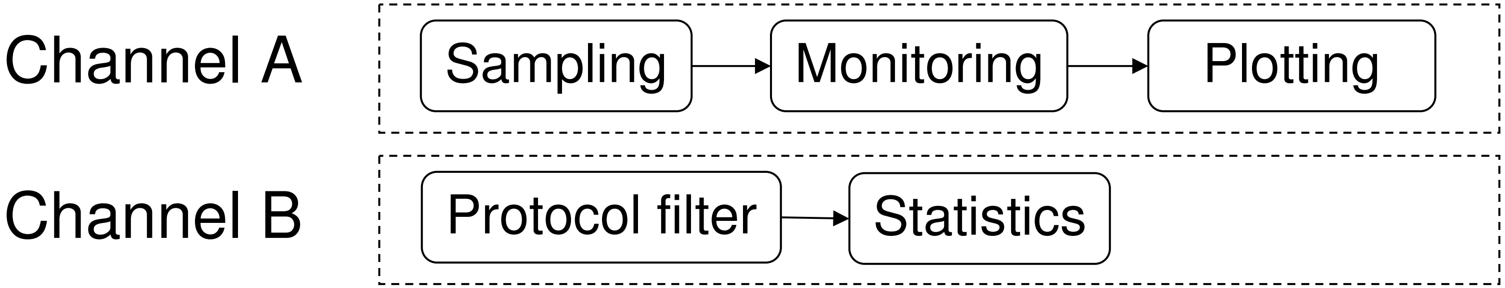
XML-based Configuration

Lightweight Modules

- *Modules*: implement well-defined functionality
  - Small building blocks for high reuse
  - Loaded at runtime on demand
  - Easily configurable (next slide)
  - Perform packet inspection ... or other tasks

  → this is where you implement your mechanisms!

- *Channels*: linear linked modules
  - Create more complex functionality



Channel A: Sampling → Monitoring → Plotting

Channel B: Protocol filter → Statistics

Flexible Configuration

- How can I configure my modules?

Module instance

Library the module is based on

```
<module name="Modules">
  <submodule name="BasicSampling"
    [illegible]>
    <configitem name="SamplingCount">200</configitem>
    <configitem name="[illegible]">[illegible]</configitem>
  </submodule>
...
</module>
```

Module parameters

Channel name

Module instantiation and configuration
→ Can use module libraries multiple times with different configuration!

```
<module name="Channels">
  <submodule name="SimpleMonitoring" stage="1">
    <configitem name="1">BasicSampling</configitem>
    [illegible]
  </submodule>
...
</module>
```

Use of the module

Channels and actual use of modules
→ Flexible grouping of small modules into larger functionality!

Communication

- Modules are leightweight, small, decoupled

  → Enables high reuse, but how can they interact?

- Data-centric communication between modules
  - Modules register for message they are interested in
    - ▶ Modules send out messages
    - ▶ Messages delivered to registered modules
  - Module: *`Hmm … interesting information I got here … maybe someone is interested in this`* → send

- Remote communication as easy as local
  - Send messages locally, remotely, or both
  - Transparent message distribution to remote Distack instances

Transparent Abstraction

- Distrack abstracts from traffic sources
  - Live traffic: buffers handle busty traffic
  - Recorded traffic: replayed with original timing
  - Simulated traffic: packet transformation for OMNeT++

- Easy and consistent packet access
  - Traffic live, replayed, or simulated … you don‘t care!
  - Easy and safe access to protocol parsers

```
TcpPacket* tcp = ippacket->getNextPacket();
if(tcp->isFlagSet(TcpPacket::TCP_FLAG_SYN))
   port = tcp->getDestport();
```

  - Supported protocols
    - ▶ Ethernet, ARP, ICMP, IPv4, IPv6, MPLS, TCP, UDP
    - ▶ More to come. Easy to implement your own!

Integration into simulations

- Few simulations of DDoS attacks and detection

In our opinion the key to understand the global and distributed behavior of DDoS attacks

- Our simulation toolkit
  - OMNeT++: time discrete simulation environment
  - INET Framework: lots of protocols (TCP, UDP, …)
  - ReaSE: topology, self-similar traffic generation, DDoS zombies

- Distack is integrated into this toolkit
  - Packet formats
    - ▶ Transparent transformation into Distacks protocol parsers
  - Time domain
    - ▶ The simulation time runs different!
  - Modules source code compatible
    - ▶ just need to recompile …

Distack is real!

Everything presented here is *running code*!

- Go and implement some modules
  - Try it out! E.g. analyze a trace file
  - Use the communication between remote instances
  - There are already over 10 modules available
- Go and do a large-scale simulation
  - Could be DDoS, could be somethings else
  - Find out how easy Distack makes your life!

Summary and Outlook

- Framework for distributed attack detection
  - Easily integrate your attack detection and traffic analysis mechanisms
  - Easy to use local and remote communication
  - Highly flexible employment
  - Transparent support for different runtime environments (e.g. simulations)

- Outlook
  - GUI support
  - More runtime environments (routers, network cards)
  - More modules to support *your* research
  - More support for large-scale simulations

Thank you! Questions?

distack
distributed attack detection

Try *Distack* now!
It`s Open Source!

www.tm.uka.de/distack