die setz ich einfach mal davor... ;)

$\boxed{\text{0 min}}$

An analysis of Internet chat systems

by

Arne Wichmann (TU München),
Christian Dewes (Prostep AG) and Anja Feldmann (TU München)

- Hi. I am Arne wichmann from TU München, Germany.
- I will present our work on chat systems which is joint work with Christian Dewes and Anja Feldmann.

$\boxed{\text{0 min}}$

Why Chat?

- Popular application, habit forming
- Computer mediated communication

- The first question is, why are chat systems at all relevant to talk about?
- After all, they do not contribute a big fraction to the internet, nor are latency or jitter requirements of chat systems problematic to fulfill.
- On the other hand they are a very popular application at least for young people, and for a number of people the use of chat systems makes up the greatest part of the time they spend using the internet.
- Even more, studies have shown that chat systems exhibit drug-like habit-forming properties.
- Moreover they are an example for computer mediated communication, which means we can study statistical properties of human behaviour with little computer intervention.
- Additionally, as they are low-bandwidth, they are interesting as a wireless application.
- In some ways SMS is already used in that way.

$\boxed{\text{1 min}}$

Why is it interesting to talk about it here?

- Ill formulated problem
  (no single Web-chat protocol, ill defined protocols, hidden in Web traffic)

Our filtering approach

- Start with well-known chat-system: IRC
- Identify properties of Web-chat
- Formulate and apply filter heuristics
- Validate filter heuristics

- So we did analyse chat systems.
- This still does not tell why it is interesting to present that here.
- The primary reason is that collecting chat traffic is far from simple.
- There is a huge number of different chat systems using different protocols.
- These protocols are usually neither documented nor well-defined, which makes catching a somewhat representative sample of chat traffic quite challenging.
- We followed the following approach: First we used IRC, which is a well-known and well-defined chat system, to identify properties of chat systems we could use to catch chat traffic.
- Then we started capturing most of the Internet traffic, and then came up with a number of heuristics to sieve chat traffic from the rest of the traffic, which we succesively refined in a number of steps.
- To make sure we catch the things we are supposed to we validated the resulting methodology.

$\boxed{\text{3 min}}$

Overview

- Types of chat systems
- Filtering approach
- Filtering Validation
- Results
- Summary and future work

- The rest of the talk will approximately follow that line.
- I will first give an overview over the types of chat systems.
- Then I will outline our approach to filter out web chat traffic, followed by the validation of that approach.
- Then I will show some preliminary results we got when capturing traffic, and at last I will give some closing remarks.

$\boxed{\text{4 min}}$

Types of Chat systems

- IRC (internet relay chat)
- Web-chat
  - HTML based
  - Applet based
- Instant messengers (ICQ, AIM, MIM)
- Others

We classified chat systems into the following types: IRC, Webchat, instant messengers, and some rest.

$\boxed{\text{4 min}}$

Types: IRC

- Widely used - relatively old
- Client/server to Server network
- Channels
- User = unique nickname
- Commands: PRIVMSG, JOIN, ISON, NICK, . . .
- IRC operators administer IRC network

- The first type, IRC, exists for over 15 years, which makes it quite old in the world of the internet.
- It is quite widely used, the 5 biggest IRC networks counting about 450.000 users on average during last july.
- It is a client server system, with the clients connetcing to a network of interconnected servers.
- Group communication is done using channels, one user can be in multiple channels.
- A user is identified using a nickname which is unique for one server network.
- There is a well-defined protocol with a sizable number of commands to for example send a message to a nick, or a channel, to join a channel, to check, if a list of nicknames are online, or to change one's nickname, or many more.
- Usually most of these aspects are handled using specialized clients, which can be quite complex applications.
- As a last aspect, the organization and administration is done by so-called IRC operators.

$\boxed{\text{6 min}}$

Types of Chat systems

- IRC (internet relay chat)
- Web-chat
  - HTML based
  - Applet based
- Instant messengers (ICQ, AIM, MIM)
- Others

The second class of chat systems are web chat systems.

$\boxed{\text{6 min}}$

Types: Web-Chat

- Widely used - newer; Simple user interface
- Client/server to single server
- Lots of systems using different protocols:
- HTML based
  - Interface: Browser
  - Protocol: HTTP
- Applet based
  - Interface: Applet window
  - Protocol: Custom or IRC

- Web chat systems are even more widely used than IRC, and, as the Web is not 15 years old, they are newer than IRC.
- The main advantage of web chat systems is a simple user interface: they use the web browser as user interface.
- The systems we have seen are client/server systems connecting to single servers.
- In contrast to IRC, there is not one Web chat system or protocol, there are a multitude.
- We divided them into two groups using one dividing characteristic: one group uses HTML as their communication protocol, the other uses HTML only to set up the connection, and after that talks a custom protocol.
- The first class uses the web browser to show all communication and, as said before uses HTTP as communication protocol.
- One could also say: they hide in HTTP.
- The second class usually creates an applet window using typically Java, and they use different custom protocols for the client-server communication.
- Or they act as frontend to another chat protocol like IRC or an instant messenger.

$\boxed{\text{9 min}}$

Types of Chat systems

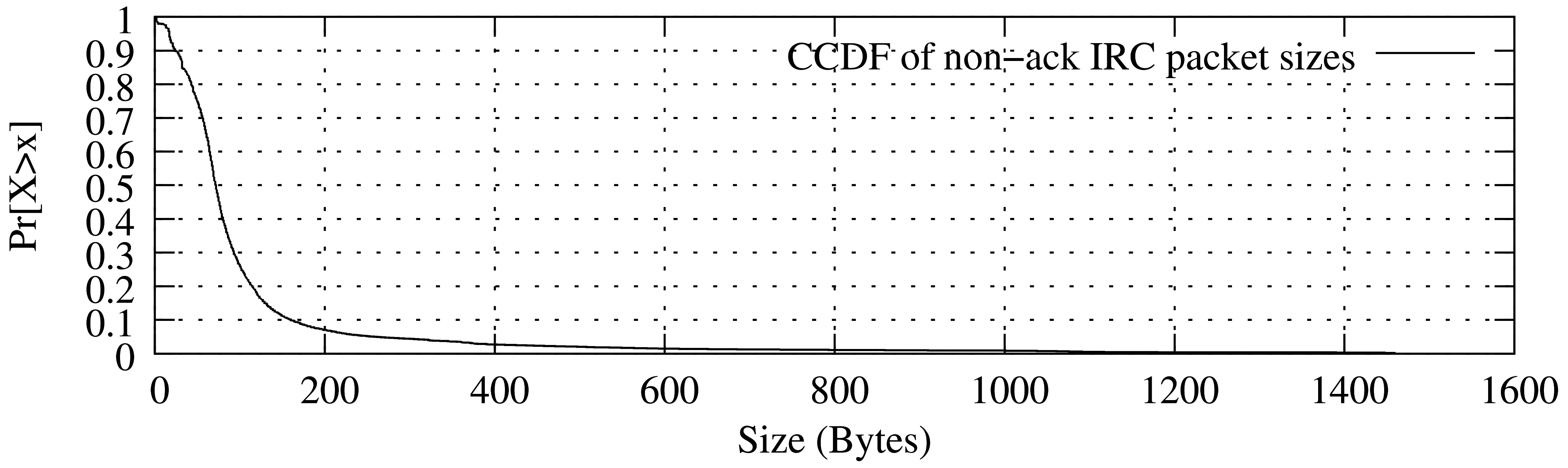
- IRC (internet relay chat)
- Web-chat
  - HTML based
  - Applet based
- Instant messengers (ICQ, AIM, MIM)
- Others

- This leads us to the next type of chat system, the so-called instant messengers.
- Examples of these are ICQ, AOL instant messenger or Microsoft Instant messenger.
- As they tend to use UDP and usually do not have a channel concept to enable group communication we did not have a closer look at these as of yet.
- This is future work.
- At last there are a number of less used different systems, for example gale, which we did not look at, too.

$\boxed{\text{9 min}}$

Approach: Properties of IRC

- Port 6667
- Well-defined protocol (RFC 1495)
- Many small packets (median $<$ 100 bytes)



- So how did we go about, as we wanted to capture web chat traffic?
- We knew that we had to look at a multitude of systems with usually unavailable protocol descriptions.
- So at first we had a look at IRC.
- Capturing IRC traffic was easy, as most IRC servers use the TCP port 6667, which is not typically used by other protocols.
- Analyzing what we found was easy, too, as IRC is documented in RFC 1495.
- Most of what we got from the analysis was general understanding as to what we should search for, but the main finding was that chat traffic should create quite small packets.
- In the graph below we plotted the probability that a non-ack IRC packet as captured by us exceeds a given size.
- We can see, that most packets are below 100 bytes, and that more than 90% of the packets are below 200 bytes.
- We used this as a starting point to capture web chat traffic.

$\boxed{\text{11 min}}$

Approach: Properties of Web-chat

- Small packets dominate

- Typical properties of HTML based Web-chat
  - Usage of suitable cache-control-headers
  - Usage of session ID's
  - Additional connections for private rooms (séparées)
  - Usage of scripting languages

- Typical properties of Applet based Web-chat
  - Usage of Java
  - Usage of an instant messenger protocol
  - Usage of IRC as underlying protocol

- After that we had a starting point we could use to capture web chat traffic.
- I will go into more details about how we did it in the next slide, but as it helps to understand what properties we found in web chat systems, I will do that first.
- The first, and single omnipresent propertiy we found is the dominance of small packets, as in IRC.
- All other properties vary significantly, although all systems have one or the other.
- HTML based chat systems typically use cache-control headers, like Pragma: no cache, cache-control: no store or cache-control no-cache.
- Others, like Cache-Control: must-revalidate, are not found, and can be used to weed out other connections that are not chat but look similar.
- Many systems use session ID's to keep state about single chat connections, many use additional connections for private rooms, also called sépar'ees, and meny use scripting languages.
- Applet based chats have a different set of typical attributes.
- Many of them use Java to set up their chat windows, and a number of them use IRC or an instant messenger protocol for their underlying communication.

$\boxed{\text{14 min}}$

Approach: Filter heuristics

- Start with large fraction of network traffic
- Group packets into bidirectional flows – retain packets
- Ignore flows dominated by large packets (50% of packets $>$ 300 bytes)
- Mark acceptable flows:
  suitable HTTP cache-control-headers, preceeding applet downloads, preceding flows using the word 'chat', . . .
- Ignore flows:
  unsuitable cache-control-headers, images, signatures of different non-chat applications, very high data rate, durations less than 30 seconds, unidirectional flows, . . .
- Accept flows which are marked and not ignored.

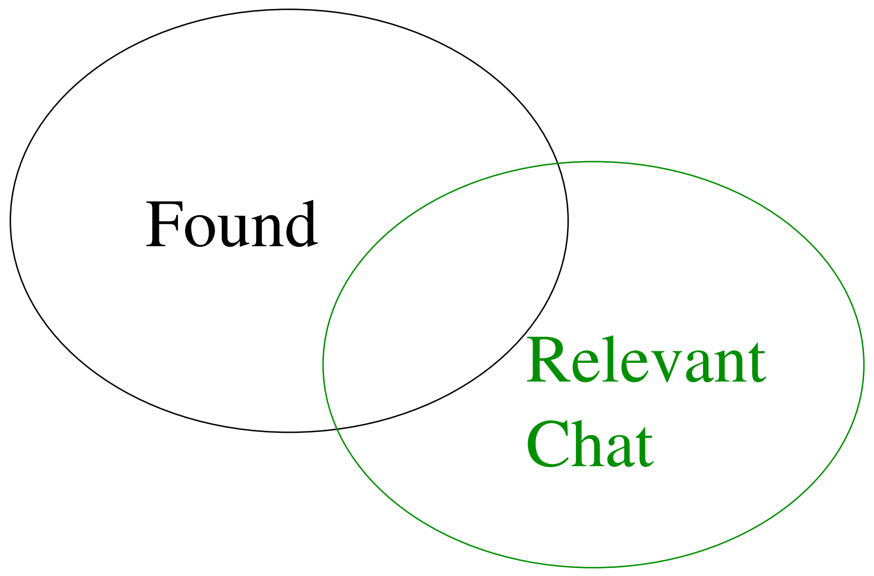
- We found and used these properties while building the filter to capture web chat traffic.
- The end result conceptually looked like follows: We first captured most TCP traffic, only ignoring ports lower than 1024 except 80, and some ports which were not used by chat systems but contained a big fraction of data, namely gnutella.
- Then we classified the packets into flows, but kept the packets.
- Then we started filtering on the flows.
- We first eliminated flows which were dominated by large packets.
- As it would have been computationally very expensive to do that on complete flows, we only did it on the start of the flows, on the first 300 packets.
- Then we continued marking flows with criteria which mede them look like Web-chat and threw out flows which had properties which should not occur in web chat flows.
- An example for the latter are images or SMTP flows.
- In the end, we saved only those flows, which were marked and not ignored.
- This strategy was quite complex and needed a lot of CPU speed.
- Moreover we could not do the postprocessing on the sniffer, as that would have caused a lot of packet loss.
- So, in reality, we used a multistage approach spread out over a number of processes on two machines.
- Details about this are in the paper.

16 min

Validation

- What percentage of Web-chat traffic do we catch (Recall)

- What percentage of what we catch is Web-chat traffic (Precision)



- So, now we have a beautiful process which captures a lot of traffic for us.
- One question remains: How good is our approach?
- We can divide this questions into two halves, which we will treat separately.
- Q1 is: What percentage of all chat traffic do we catch? and Q2 is: What percentage of what we catch is chat traffic?
- We borrowed this approach from Information Retrieval, and we will call the two values as they do: Recall and precision.

$\boxed{\text{17 min}}$

Validation: Bounding Recall

$$RECALL = \frac{|RELEVANT \bigcap FOUND|}{|RELEVANT|} = 1 - \frac{|MISSED|}{|RELEVANT|}$$

- lower bound for Relevant
  - Identify likely Web-chat connections: Trace traffic to known Web-chat servers
  - Ignore short connections ($< 30s$)
  - Check for Web-chat: manual testing
- upper bound for Missed
  - Apply methodology to above trace
  - Find connections that are Web-chat but not identified

- So let's look at the first Question about Recall.
- We can formulate Recall precisely as the number of relevant and found items divided by the number of relevant items.
- Or, silghtly reformulated, as one minus the number of items we missed divided by the number of relevant items.
- So, if we want to find a lower bound for recall, we need a lower bound for the number of relevant items and an upper bound for the missed items.
- How do we find a lower bound for the relevant items?
- We use a trace which contains traffic to known web chat servers.
- In this trace we ignore all connections shorter that $30s$, as this is about as much as the time needed to set up the chat connection.
- Then we manually check the remaining connections if they contain web chat.
- To find an upper bound for the missed items, we apply our set of filters to the above trace, and find those connections which are web chat but are not in the resulting trace.

$\boxed{\text{19 min}}$

Validation: Results of Recall

- SELCHAT*: trace to known web-chat systems
- $>$ 537 Web-chat connections in SELCHAT* ($\leq RELEVANT$)
- WEBCHAT2: trace using our methodology
- 532 connections in SELCHAT* not in WEBCHAT2
- 44 Web-chat connections ($\geq MISSED$)
- $RECALL > 1 - \frac{44}{537} \approx 91.7\%$

- So, what did we find?

- We did a 4 day trace to 35 known Web-chat servers, weeded out all connections shorter than $30s$, and checked that more than 537 connections were web chat connections.

- This is what we used as an lower bound for RELEVANT.

- We did a trace using our methodology at the same time, and looked which connections were in the first trace but not in the second, and found 532 connections.

- After manually checking these connections only 44 connections remained which really were web chat.

- This we use as a lower bound for MISSED.

- The overall recall can thous be bounded to 91.7%

$\boxed{\text{21 min}}$

Validation: prob. Bound for Precision

$$PRECISION = \frac{|RELEVANT \bigcap FOUND|}{|FOUND|} = 1 - \frac{|WRONG|}{|FOUND|}$$

- Validate that connection is Web-chat $\Rightarrow$ manual testing
- Manual testing for > 1500 connections $\Rightarrow$ sampling
- Different classes $\Rightarrow$ stratified sampling
- Stratified sampling
  - classes based on different properties of Web-chat systems
  - precision estimated for each class
  - use weighted averages

- The second question is about Precision.
- Precision is the relation of the number of relevant found items to the number of found items.
- Or, put differently, one minus the number of wrongly found items divided by the number of found items.
- To get an estimate for the number of wrongly found items, we manually checked the found connections of the Webchat 2 trace as on the last slide.
- As these connections amounted to more than 1500 we only took samples of these connections.
- But, as we wanted to ensure that we had covered all different types of web chat systems, we took these samples from each class individually and then used a weighted average based on the number of instances in each class.
- This is called stratified sampling.

$\boxed{\text{22 min}}$

Validation: Results of Precision

- 1514 connections in WEBCHAT2
- 391 connections checked manually
- $PRECISION \approx 93.1\%$

After manually checking 391 of the 1514 connections in our test trace we found a Presision of 93.2% based on the weighted average.

soll die tabelle mit rein?

$\boxed{\text{23 min}}$

Results: Traces

- Preliminary results
  - only one location
  - specialized location (campus Saarland University)
  - limited time period (one week)
- Trace Overview
  - 12/13/2002, 15:46 MET, until 12/21/2002, 15:46 MET
  - 950GB filtered, 238MB reduced $\cong 5 \cdot 10^9$ packets, 0.032% drop rate.
  - IRC data – same trace: 192$MB \cong 4 \cdot 10^6$ packets.

- After developing our methodology and validating it, we captured a packet trace using thet methodology, and looked at the resulting characteristics.

- These characteristics should be seen as preliminary results only, as they ware taken at only one location, moreover a quite specialized one, as it is from a university campus, where people act differently than at home.

- And the traces only cover one week, which is a rather limited time period.

- The trace was taken at the Internet connection of Saarland University, in December 2002.

- We filtered 950GB and reduced them to 238MB or 5 Billion packets.

- The drop rate was 0.0032%.

- An IRC trace was taken at the same time and resulted in 192 MB or 4 Million packets.

$\boxed{\text{24 min}}$

Results: Overview

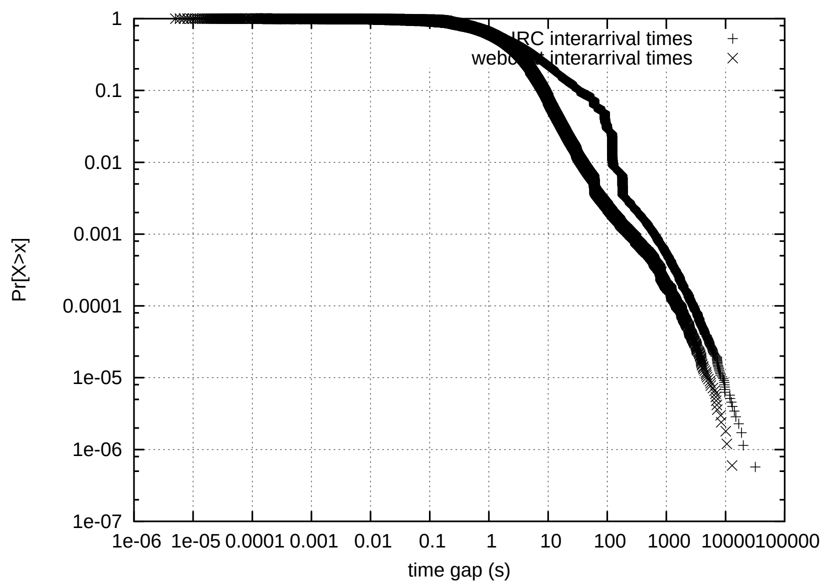
- about 10 times as many bytes received as sent, varying wildly
- apparently exponentially distributed session interarrivals
- session durations vary significantly
- packet interarrival times and packet size distributions, see following slides

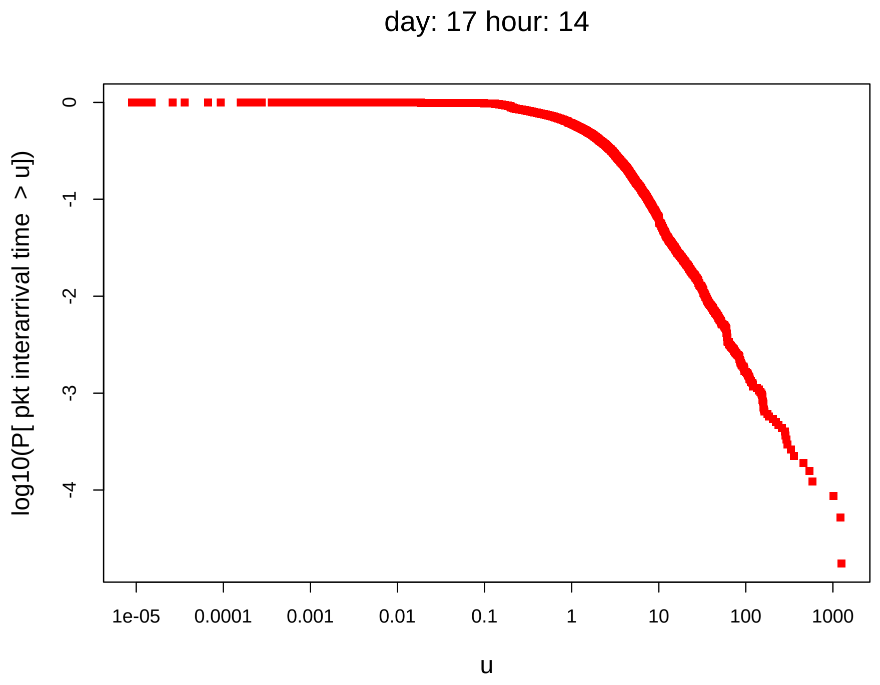
- The traffic characteristics we found looked as follows.
- The relation of received to sent bytes varies wildly from about 1 to 100, with a center around 10, for both IRC and Webchat.
- Session interarrivals seem to be exponentially distributed, as we expected.
- Session durations vary significantly, apparently more than exponentially, but we can not claim that they are heavy-tailed.
- We will have a look at packet interarrival times and packet size distributions at the next slide.

ich sollte folien für die restlichen ergebnisse in reserve haben

$\boxed{\text{26 min}}$

Packet interarrival times





CCDF of packet interarrival times for IRC without PINGs and Web-chat

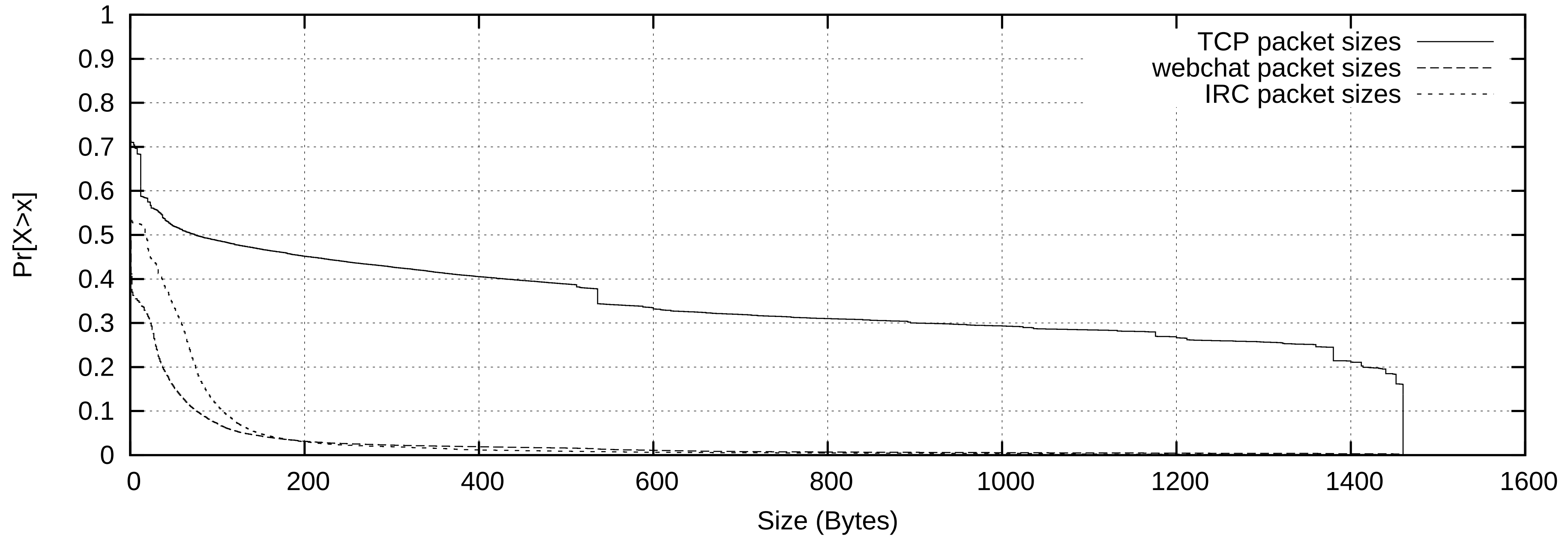
CCDF of packet interarrival times for Web-chat during 2pm to 3pm on 12/17/2002

Packet interarrival times seem to be heavy-tailed

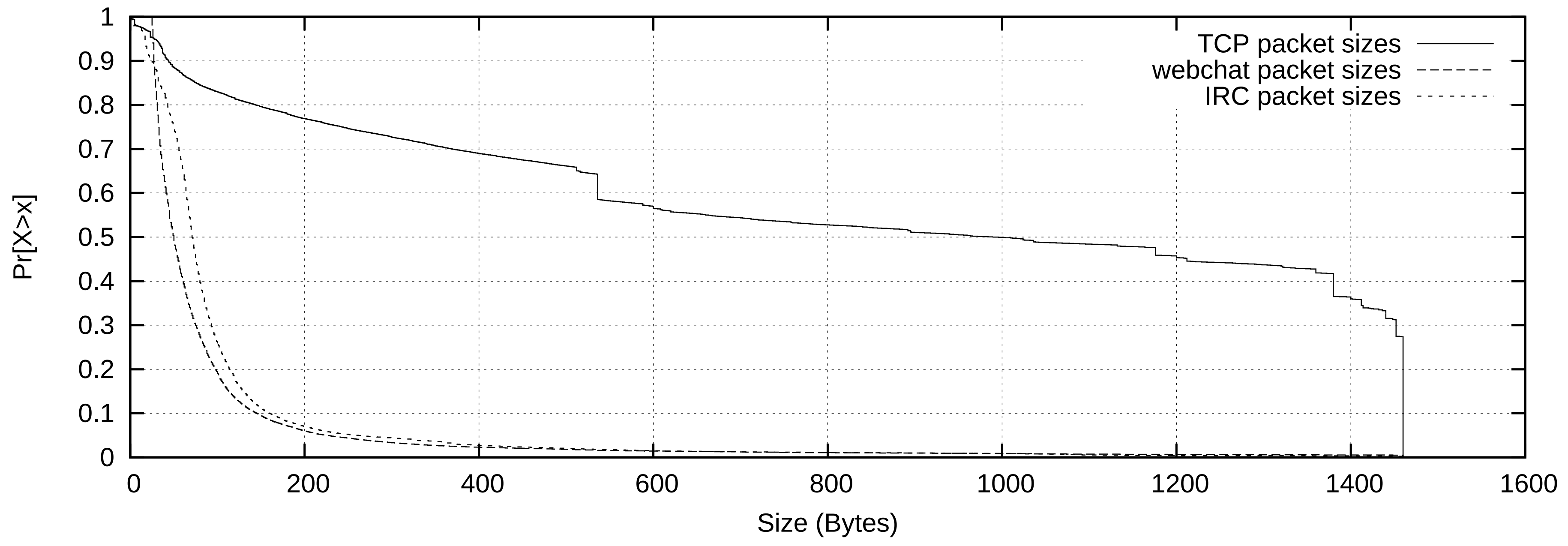
- On the left graph we can see a complemtary cumulative distribution function of the packet interarrival times for IRC and web chat over the whole week of traces.
- In the case of IRC we removed packets which contained a ping command or reply from the traces, as this influences the graph heavily.
- PINGs are usually sent out by an IRC server, if a client does not send anything for two minutes.
- On the x-axis we can see the time gap between two packets, in seconds, on the y-axis we can see the probability that a time gap is bigger than a given value.
- Both axes are logarithmic.
- Both graphs we can see exhibit variations over a large timescale which can be interpreted as evidence for a heavy-tailed distribution.
- The bend in the IRC-graph seems to stem from clients using the ISON-command to automatically check presence of a list of other users or from IRC-automatons, so-called bots, sending messages in regular intervals.
- The other graph shows a typical example of a CCDF-plot of packet interarrival times of one hour of web-chat traffic.
- The axes are like in the other plot.
- Here we can see the straight line even clearer.
- Also we can expect the distribution over an hour to have better stationary qualities.
- This, too, is an evidence for heavy-tailedness.

$\boxed{\text{29 min}}$

Packet size distributions



CDF of packet size distributions including ACKs



CDF of packet size distributions excluding ACKs

Typical packet sizes for Web-chat and IRC are below 100 bytes

- Another interesting property, especially as we used it in capturing, is the packet size ditribution of chat packets.
- Here we have two CDF-plots of the packet sizes of TCP, Web-chat and IRC-packets, including and excluding ACKs, respectively.
- On the x-axis we see the size of the packets, in bytes, and on the y-axis we see the probability that a packet is bigger than the respective size.
- As expected, we can see that web-chat-packets exhibit similar size characteristics as IRC-packets do.
- They generally even are a bit smaller.
- As a contrast we put general TCP packet sizes from an earlier trace into the picture.
- There we can see, that 50% of all non-ack TCP-packets are larger than 1 kilobyte, and that 70% are larger that 300 bytes.
- This shows once more, that the packet size distribution is a quite good criterium to filter out most non-chat traffic.

31 min

Summary

- A new method to catch traffic to Web-chat systems.
- A first stab at analysis of Web-chat traffic data

Future Work

- Improve precision and recall to $> 95\%$
- Traces from more diverse places over a longer time
- Gain deeper understanding of the origins of the distributions of session durations and packet interarrival times

- This is what we did.
- We developed, applied and validated a new method to filter traffic to web chat systems.
- The first small application was to collect enough data to do a first traffic analysis.
- This work is by no means finished, there are a number of things which can and should still be done.
- The first and most straightforward point is to improve the quality of the filter, or at least make some claims about how the misses in the filter impact the quality of the results we get.
- The second point is to find some more and better places to capture web chat traffic, and to take traces over a longer time to get more significant results.
- A different direction would be to get some deeper insights where these heavy tails come from, but I think I should ask someone with knowledge about humans about that.

32 min