Recap: Tracking Anonymous Peer-to-Peer VoIP Calls on the Internet

Scott E. Coull and Amos Wetherbee

April 7, 2006

Encoding a bit

1. Packet flow of *n* bits
2. Select *2r* packets from the first *n-d* packets at random
   - *d* is used to prevent overflowing the packet flow
   - Hence, $d \leq \left\lfloor \frac{n}{2} \right\rfloor$

Encoding a bit

3. Pair each chosen packet with a packet in the flow *d* packets later
4. Compute the Inter-Packet Delay for each pair:

$$ipd_{z_k,d} = t_{z_k+d} - t_{z_k}$$

- Gives us the timing difference between the k^{th} packet chosen at random and the $(k+d)^{th}$

Encoding a bit

- Random variables for IPD are independently and identically distributed (*i.i.d.*)
  - We select packets independently at random (independence)
  - Packets are taken from the same arrival distribution (identically distributed)

Encoding a bit

5. Split the IPDs into two groups of equal size
   - Since our IPDs are *i.i.d.* we can expect these groups are similar w.r.t. mean and variance
6. Calculate the midpoint difference between corresponding IPDs across the groups

$$Y_{k,d} = \frac{ipd_{1,k,d} - ipd_{2,k,d}}{2}$$

Encoding a bit

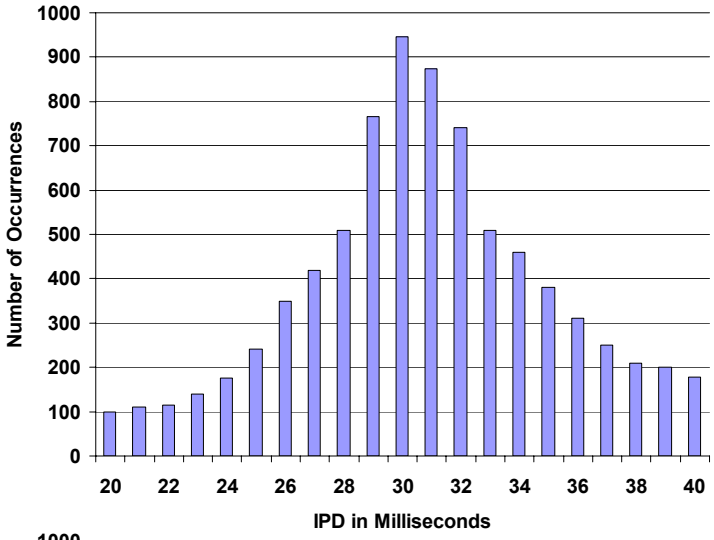
7. Compute the average of the midpoint differences

$$\overline{Y_{r,d}} = \frac{1}{r}\sum_{k=1}^{r} Y_{k,d}$$

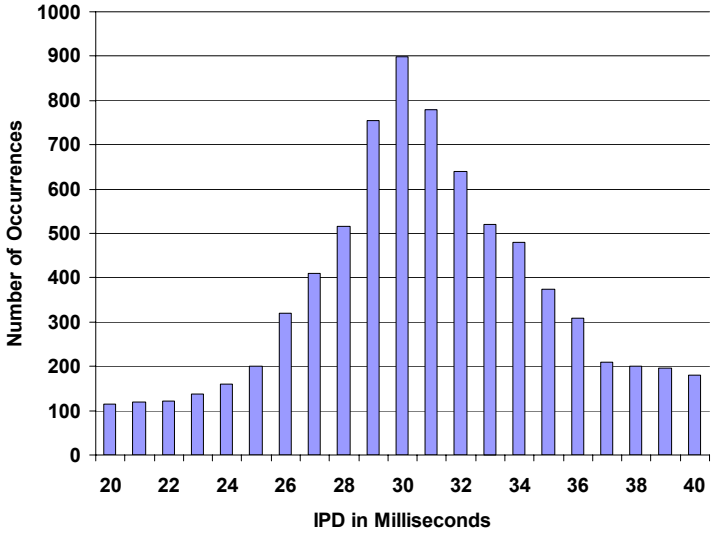
- We want to change this value to be skewed by *a*
- If we increase $ipd_{1,k,d}$ and decrease $ipd_{2,k,d}$ we skew the average positively

Encoding a bit: An Example

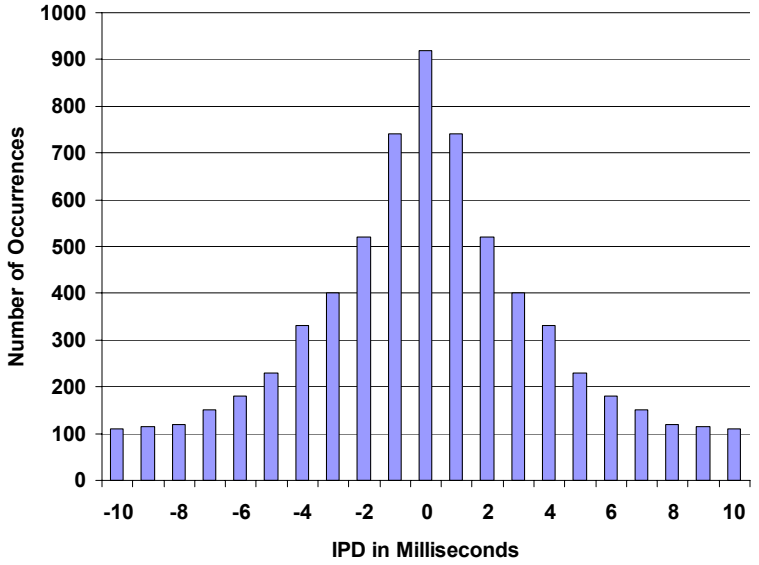
$IPD_{1,k,d}$



$IPD_{2,k,d}$

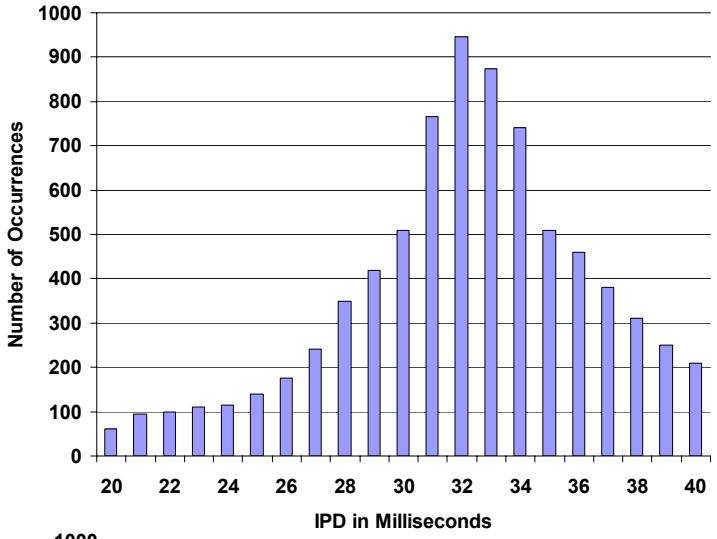


$\overline{Y_{r,d}}$

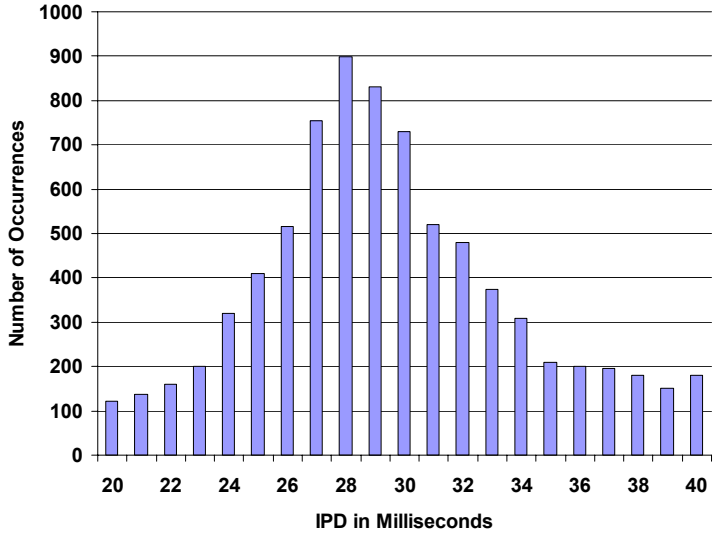


Encoding a bit: An Example

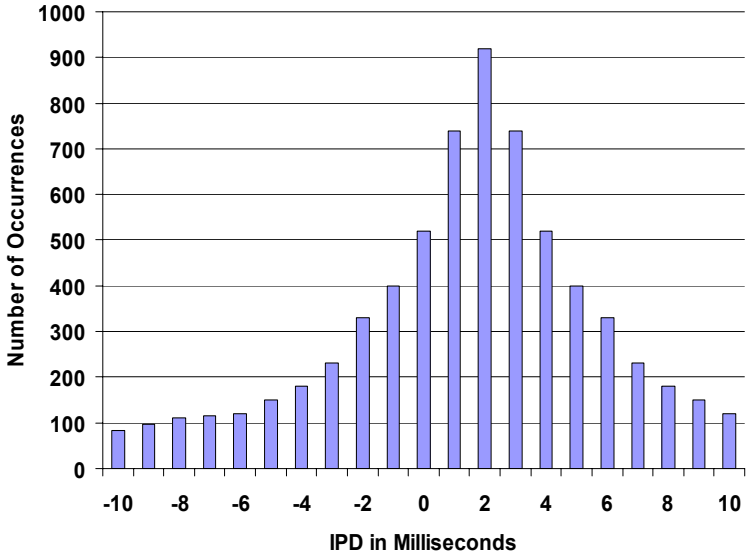
$IPD_{1,k,d}$



$IPD_{2,k,d}$



$\overline{Y_{r,d}}$



Reactions: Tracking Anonymous Peer-to-Peer VoIP Calls on the Internet

Scott E. Coull and Amos Wetherbee

April 7, 2006

Questions

- What is the value *d*?
  - Ensures that we do not overflow the number of packets we have when creating pairings
- What is the value *r*?
  - The number of packets to alter
  - The larger the value, the better the chance of recreating the proper distributions
    - As *r* increases, we can expect more of them to arrive with little or no jitter
  - Also reduces channel capacity as it increases

Questions

- How does Skype route calls?
  - In some cases Supernodes route traffic, in others it is direct peer-to-peer
  - Refer to:
    - **An Experimental Study of the Skype Peer-to-Peer VoIP System**

      (http://iptps06.cs.ucsb.edu/papers/Guha-skype06.pdf)
    - **Silver Needle in the Skype**

      (http://www.secdev.org/conf/skype_BHEU06.pdf)

Questions

- How can we ***reduce*** the delay of a packet?!
  - Slow all VoIP packets down by *a* time then allow the ‘reduced delay’ packets to proceed without that delay
- Can’t we just check for the same encrypted traffic on both ends?
  - Yes – See Detecting Stepping Stones by Zhang & Paxson
  - Active monitoring is necessary because it isn’t easy to distinguish VoIP flows

Questions

- How do you encode multiple bits into a call?
  - Good question; they aren't specific
  - My guess:
    - Think of *n* as a window size
    - Utilize multiple windows of size *n*
  - In their application this is unnecessary

Comments

The paper has a fairly good mixture of mathematical reasoning, charts and diagrams, and experimental results.

…I really like the idea of a watermark that is built upon the expectation of something and the central limit theorem.

When I first read this paper, I almost felt satisfied by the level of analysis performed…But then I read "Capacity Estimation and Auditability of Network Covert Channels": a work done over 10 years ago.

Comments

- Paper fails in defining a practical, real-time algorithm
  - How do we buffer the packets?
    - In fact, the authors claim no buffering is needed
  - How can we find the average difference without storing all packets for the call first?
  - One possibility:
    - Perform ‘addition’ and ‘subtraction’ of delay on per packet basis rather than computing the entire distribution

Extensions

- Anonymizing networks that add/subtract jitter to inter-arrival times
  - Subtracting jitter (i.e. maintaining QoS) is equivalent to quantizing the transmission rate
  - Adding jitter is equivalent to randomizing the distribution
  - Both can work, but one is better
  - Could these work for VoIP? At what point would the call quality suffer?

Extensions

- Changes to the watermarking system to accommodate shorter/longer calls more efficiently
  - Can we choose an optimal technique based on the call length/type?
- Alternate method of watermarking by ‘avoiding’ a specific IPD value
  - Use in traceback mechanisms?

Extensions

- Leaking RFID key by car engine RPM level at idle
- Leaking information through CRT monitor refresh rate
- Angular velocity of a Blu-Ray DVD…
- <Insert ridiculous source of covert channel here>

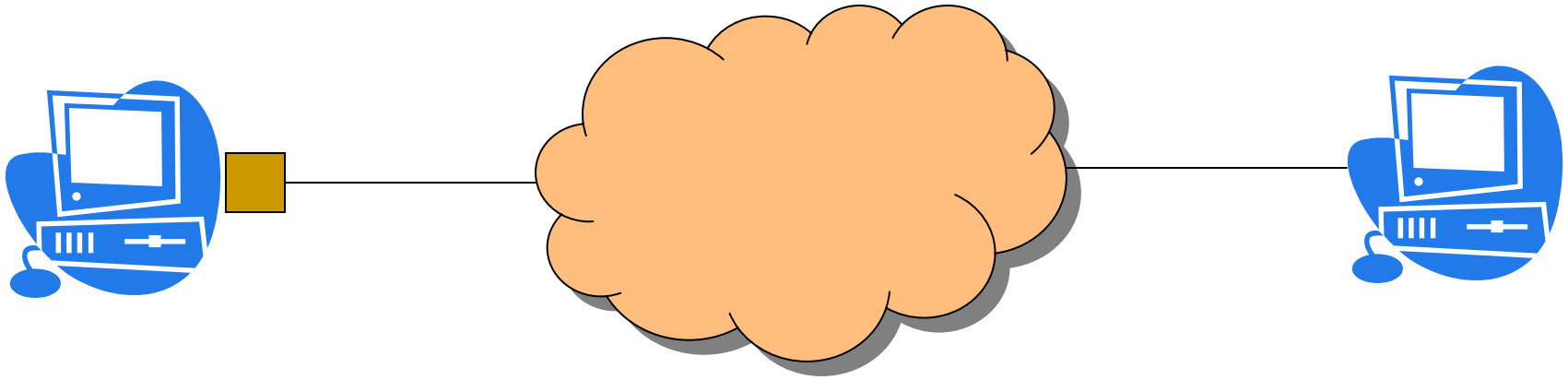
- Covert channels are everywhere
  - Still lots of interesting research to be done

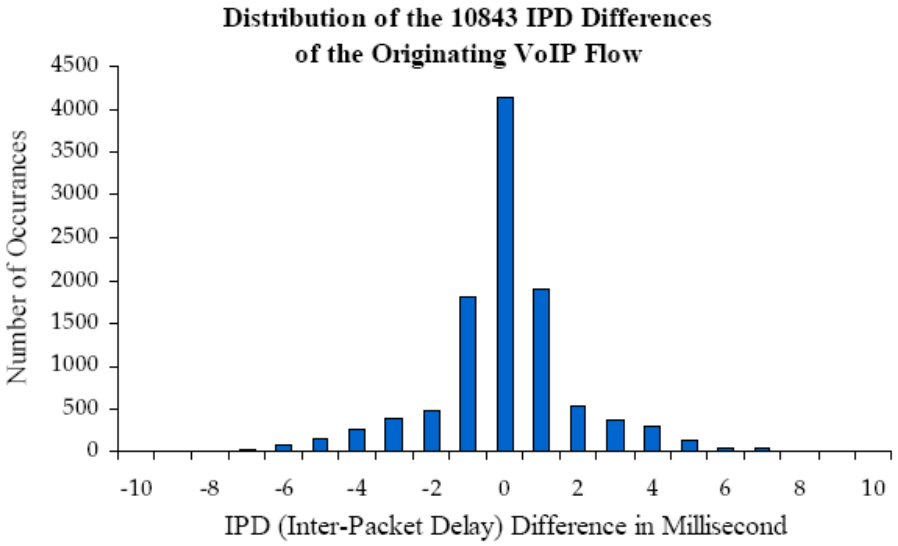
Covert Timing Channels and their Defenses

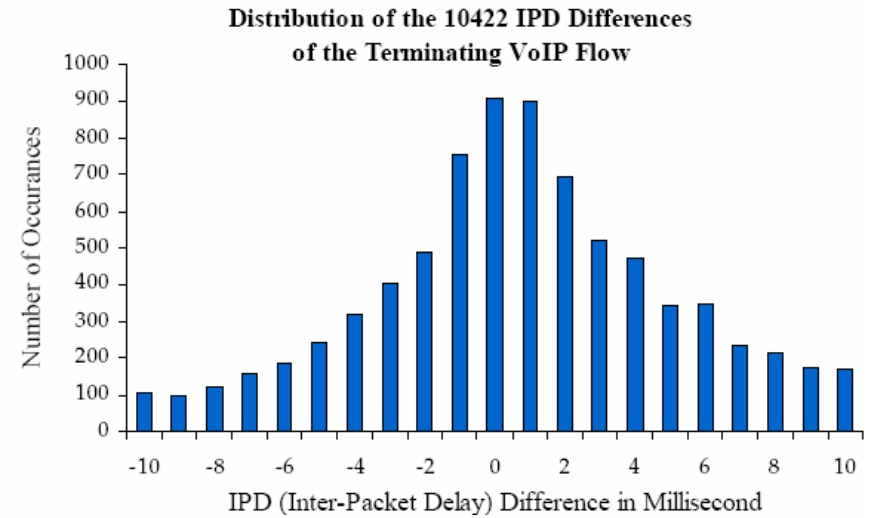
Scott E. Coull

April 7, 2006

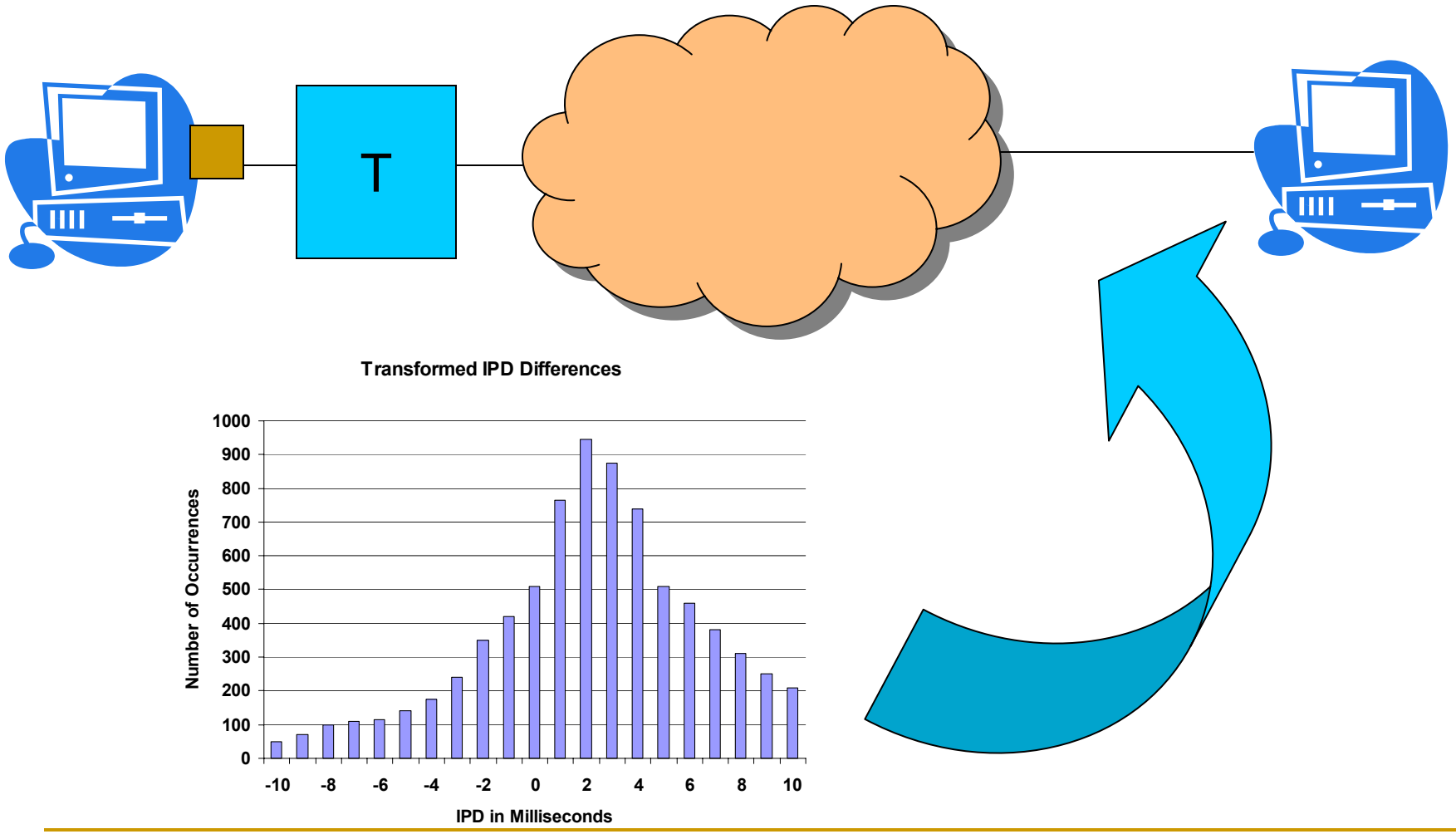
Revisiting VoIP Tracking



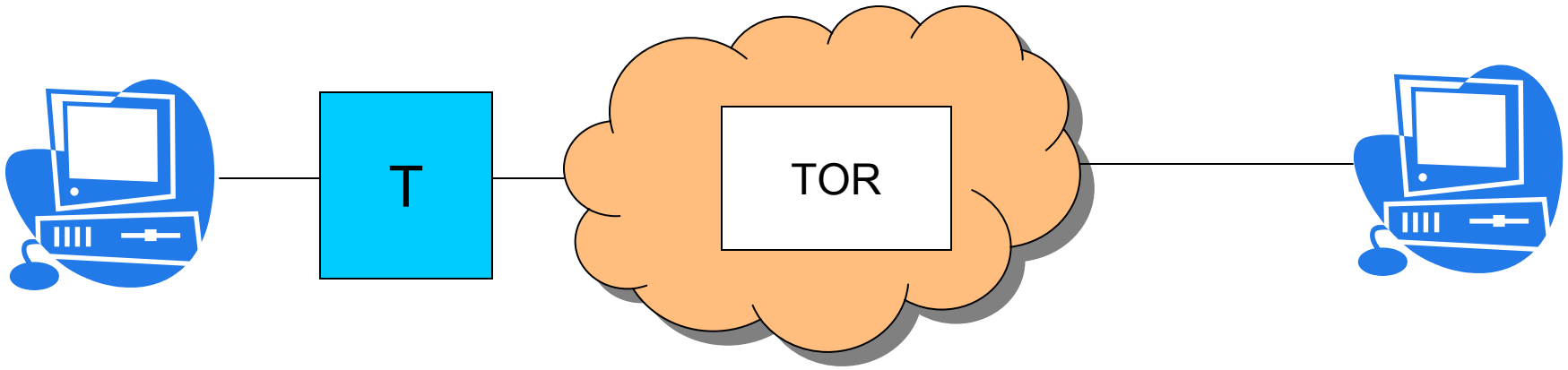




Revisiting VoIP Tracking

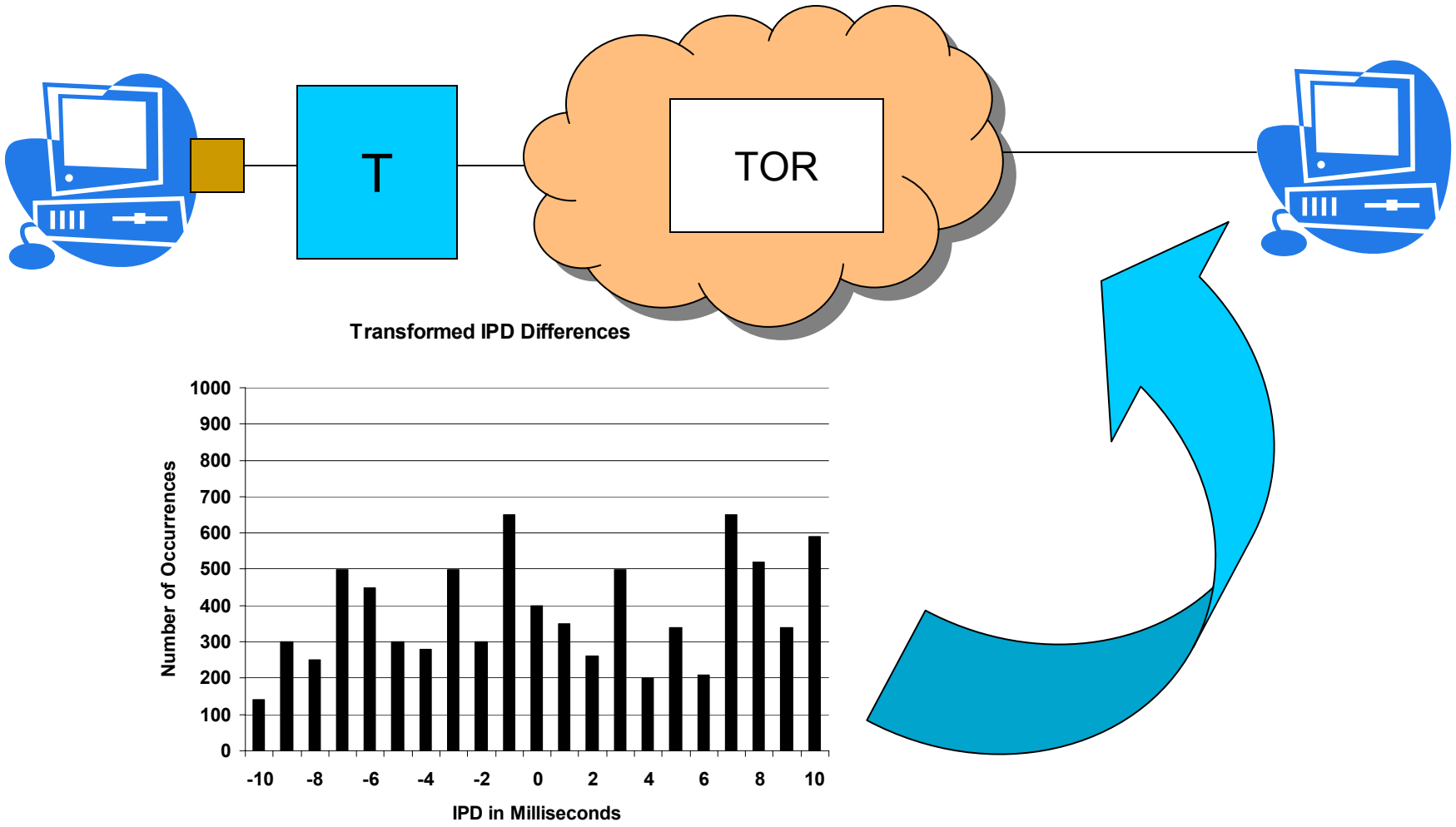


Naïve Method of Defense for VoIP



- Sender increases rate of sending to 20ms
  - Makes up for the delay introduced by TOR
- T adds 2ms delay to encode a ‘1’ bit
- Sender chooses some random number of TOR routers to send the packet through
  - Thereby introducing ‘random’ delay after T

Naïve Method of Defense for VoIP



Generalizing the Defense

- ‘Randomize’ the timing
  - Fuzzy Timing – Wei-Ming Hu
  - Network Pumps – Kang, Moskowitz, Lee
  - Jammers – Giles and Hajek
- Detect changes to variance in inter-arrival times
  - Detecting IP Timing Channels – Cabuk, Brodley, Shields

Fuzzy Timing

- Implemented in VAX security kernel
- Software timing channels are easy to audit by the Trusted Computing Base (TCB)
  - Randomize timing of scheduled processes
  - Check for known modulation techniques
- Hardware timing channels are more tricky
  - Bus-Contention as timing channel
  - Not auditable and not under control of the TCB

Fuzzy Timing

- Solution to hardware control problem:
  - Add noise to all timing information throughout the system
- Need to address clock and I/O interrupts

Fuzzy Timing

- For the system clock:
  - Set a counter to a random value
  - Increment the counter at every 1 microsecond
  - Produce an interrupt when the counter overflows
  - Accurate system time is kept separately by adding the number of increments, and it updated when the interrupt occurs

Fuzzy Timing

- For the I/O clock:
  - Need to consider the time the event occurred (downticks) and the time the notification interrupt is sent (upticks)
  - Time is ‘fuzzed’ between these two ticks by a uniformly distributed random variable

Fuzzy Timing

- Fuzzy timing effects:
  - Reduced the channel bandwidth by ‘two orders of magnitude’
  - Resultant bandwidth of less than 10 bits per second (?)
    - No need to audit the channel
  - Makes it difficult to do any timing attacks within the host, including software timing attacks

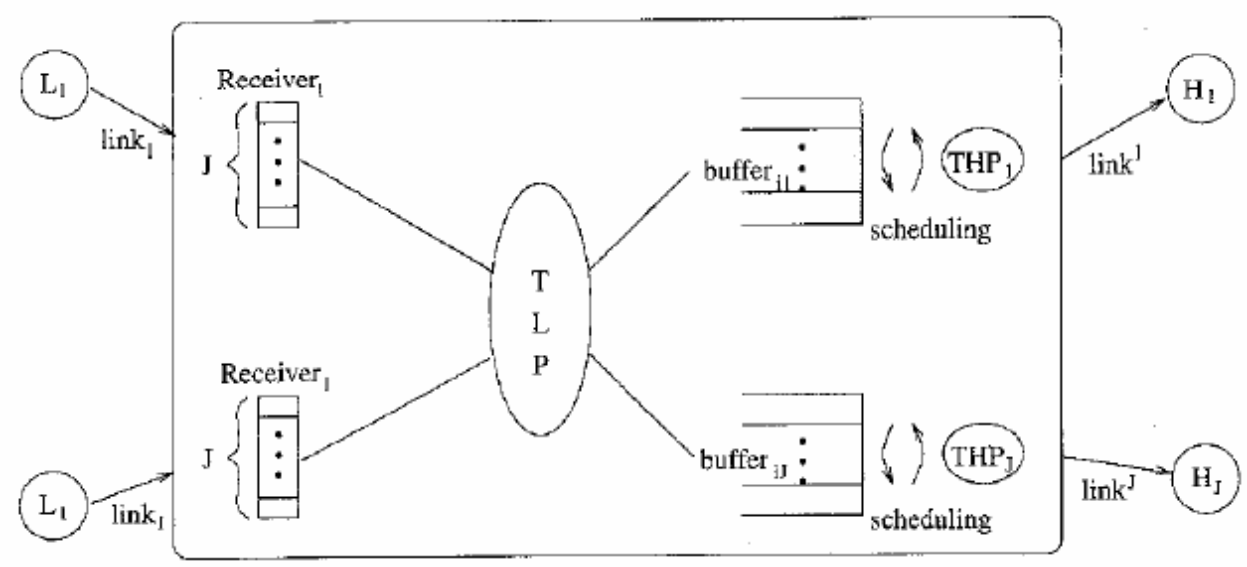
Network Pumps

- Multi-level Secure (MLS) Network:
  - High level that contains sensitive information
    - Only members of the high level can access information within the high level
  - Low level that contains information for all users
    - All members, both low level and high level can access this information
- When a high level user gets low level information, they can modulate ACK timing

Network Pumps

- The Pump
  - An intermediary between high and low level that intercepts messages from low to high and ACKs from high to low

Network Pumps



- How it works:
  - Lows (L_i) sends to their Receiver queue
  - Trusted Low Process (TLP) takes message from Receiver queue and routes it to the proper buffer
    - ACK is sent by Pump to L_i when message is placed in buffer after a random delay
  - Trusted High Process (THP) delivers the message to Highs (H_i)

Network Pumps

- Some considerations:
  - The Pump acts as a router so queuing is important
    - Design allows for max-min fair queuing strategy
  - Throughput must also remain unhindered
    - ACK rate is tied to the retrieval rate of the server from its buffers

Network Pumps

- Some considerations:
  - Effects on covert channels
    - Number of possible ACK timings define the channel capacity
      - High can still affect timing by quickly removing things from its buffer
    - These times are restricted by the ACK randomization and the queue size
    - Capacity of the channel can be arbitrarily restricted if we know the overhead time

Jammers

- Any device that limits the capacity of the covert timing channels
  - Typically implemented by delaying the communications by a random amount
  - A Network Pump, for instance
  - Our naïve VoIP defense

Jammers

- Develop optimal jamming strategies to prevent information leakage
- Develop optimal coding strategies that maximize information leakage through a jammer
- Find bounds on the channel capacity

Jammers

- Jammer constraints:
  - Any delay strategy can be used, but no deletion or insertion of packets
  - Maximum-Delay-Constrained (MDC) jammers
    - Delays no packet for more than *D* time
  - Maximum-Buffer-Constrained (MBC) jammers
    - Holds no more than *B* packets in its buffer
  - Average-Delay-Constrained (ADC) jammers
    - Average over all D_i packets delays is no more than *D*

Jammers

- Coder constraints:
  - Coder and decoder are aware of delay and the amount
  - Coder and decoder are not aware of the jamming strategy
  - Coder does not receive feedback from the decoder

A Short Information Theory Aside

- Mutual information:
  - Measures how much information two distributions share

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \frac{p(x,y)}{f(x)\, g(y)},$$

  - *p(x,y)* - probability when we combine the values from X and Y into a single distribution
  - *f(x)* - probability of value x in X
  - *g(y)* - probability of value y in Y

A Short Information Theory Aside

- Divergence (or Kullback-Leibler Distance):
  - Measures how 'different' two distributions are

$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

  - Generalized case of mutual information
  - Not a 'true' distance because it does not satisfy triangle inequality
    - Also, it is not symmetric

Jammers

- Problem formulation:
  - Consider mutual information rate between the input and output distributions

$$I(X;Y)=H(X)+H(Y)-H(X,Y)$$

    - If X and Y have exactly the same distribution:

$$I(X;Y)=H(X)=H(Y)$$

    - If X and Y are completely independent:

$$I(X;Y)=0$$

Jammers

- Problem formulation:
  - Consider a zero sum game between encoder and jammer
    - The coder wants the input to look as close to the output as possible
    - The jammer wants the input to look nothing at all like the input

Jammers

- Jammer strategies:
  - Make output stream appear random
    - Increase entropy of output conditioned on input
    - Results suggest this does not work very well
  - Quantize output levels and discretize output timing
    - Reduces the entropy of the output distribution
    - Such methods appear very useful

Jammers

- Coder strategies:
  - Make input process appear random
    - Increases the entropy of the input process
    - Provides adequate, but not optimal performance
  - Take advantage of delay constraints
    - Decreases the entropy of the output conditioned on input
    - Provides the best results

Jammers

- Channel Capacity Results:

Jammers \ Delay (Buffer Size)	D=2 ms (B=20 packets)		D=20 ms (B=200 packets)		D=200 ms (B=2,000 packets)		D=2,000 ms (B=20,000 packets)	
	UB	LB	UB	LB	UB	LB	UB	LB
No Jammer	9183 bps							
Maximum-delay-constrained	1985 bps	1073 bps	358 bps	188 bps	52 bps	28 bps	6.9 bps	3.7 bps
Maximum-buffer-constrained	875 bps	437 bps	91 bps	45 bps	9.2 bps	4.5 bps	0.92 bps	0.45 bps
Average-delay-constrained (λ known)	2066 bps	138 bps	207 bps	13.8 bps	20.7 bps	1.38 bps	2.07 bps	0.138 bps
Average-delay-constrained (λ unknown)	1985 bps	138 bps	358 bps	13.8 bps	52 bps	1.38 bps	6.9 bps	0.138 bps

Detecting IP Timing Channels

- IP covert timing channels rely on timing interval to synchronize and send information
  - Would such timing channels exhibit abnormal inter-arrival behavior?
  - Inter-arrival timing should be more consistent for timing channels

Detecting IP Timing Channels

- Technique 1: Utilize variance of the inter-arrival times in the distribution
  - Create window of *w* packets
  - For each window, compute the standard deviation
  - Calculate the pairwise differences between std. deviations
  - Compute std. deviation of these pairwise differences

$$regularity = STDEV\left(\frac{|\sigma_i - \sigma_j|}{\sigma_i}, i < j, \forall i, j\right)$$

Detecting IP Timing Channels

- Technique 2: ε-Similarity
  - Compute the relative difference between adjacent inter-arrival times

$$\frac{\left|P_i - P_{i+1}\right|}{P_i}$$

  - Find the percentage of differences that are less than some ε

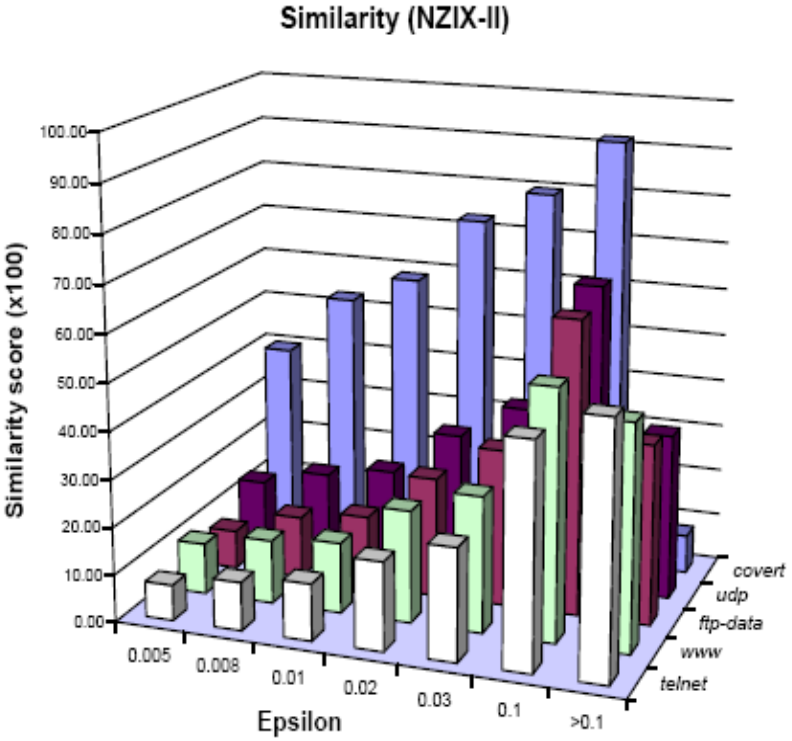
Detecting IP Timing Channels

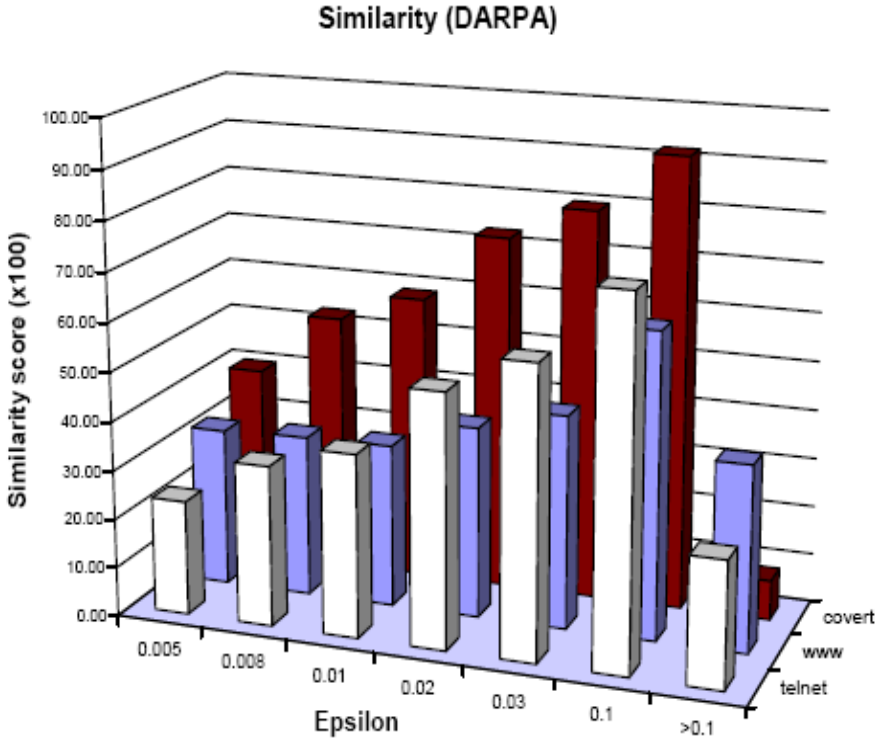
- All tests run on NZIX-II and DARPA '99 datasets
- Test case 1: A simple timing channel
  - Fixed timing interval of 0.04 seconds
  - Regularity measure of variance:

Dataset	Application	w=250	w=100
NZIX-II	WWW	22.14	34.32
NZIX-II	FTPD	7.77	16.46
NZIX-II	TELNET	12.08	18.15
NZIX-II	UDP	16.57	27.18
DARPA	WWW	21.59	62.32
DARPA	TELNET	17.70	52.21
	COVERT-I	2.18	4.63

Detecting IP Timing Channels

- Test case 1: A simple timing channel
  - ε-Similarity measure of relative difference:





Detecting IP Timing Channels

- Test case 2: Varying the timing interval
  - Attacker varies the timing interval as a countermeasure to increase the variance
  - Alternate between 0.04, 0.06, and 0.08 second intervals
    - Done in a random, or sequential fashion
  - Regularity measure fails
    - The rate at which the intervals switch must be less than the window size
      - We need to see all intervals in a single window to get the variance

Detecting IP Timing Channels

- Test case 2: Varying the timing interval
  - ε-Similarity still works
  - Scores when rotating intervals in sequential and random order:

Method	t	ϵ-Similarity Score 0.005	0.008	0.01	0.02	0.03	0.1	>0.1
Sequential	250	34.17	45.17	51.23	67.38	75.29	90.75	9.25
	100	34.12	45.77	52.78	67.53	75.54	90.50	9.50
	50	34.22	46.87	53.68	67.68	75.09	89.89	10.11
	10	34.87	46.37	51.83	67.58	76.19	90.65	9.35
Random	250	36.51	48.02	53.47	68.30	76.20	90.49	9.51
	10	35.21	46.88	52.55	68.29	75.67	90.28	9.72
Original		39.92	52.83	58.58	72.79	79.74	91.85	8.15

Detecting IP Timing Channels

- Test case 3: Injecting noise
  - Attacker injects noise from other protocols into the timing channel to add variance
  - Regularity measure fails again for similar reasons
  - ε-Similarity approaches actual traffic as noise increases
  - Note that as noise increases, channel capacity decreases

Summary

- How do we reduce covert channel capacity?
  - Add lots of noise through randomization or quantization
- In most cases it is impossible to completely close covert channels
  - We can reduce the capacity to a suitably low amount
- Encoding of information in network traffic necessarily deviates the variance
  - This allows us to better audit such channels

References

S. Cabuk, C. Brodley, and C. Shields. IP Covert Timing Channels: Design and Detection. In Proceedings of the ACM Conference on Computer and Communications Security '04. October, 2004.

J. Giles and B. Hajek. An Information-Theoretic and Game-Theoretic Study of Timing Channels. In IEEE Transactions on Information Theory, 48(9). September, 2002.

W. Hu. Reducing Timing Channels with Fuzzy Time. In Proceedings of IEEE Computer Society on Research in Security and Privacy. May, 1991.

M. H. Kang, I. S. Moskowitz, and D. Lee. A Network Pump. In IEEE Transactions on Software Engineering, 22(5). May, 1996.