Worms & Botnets

Otto J. Anshus
UiT & UiO

Structure

- THE Classic Worm paper
- Botnets

Worm Programs - Early Experience with a Distributed Computation

Shoch and Hupp
Xerox Palo Alto Research Center

Communicatons of the ACM

1982

What is the paper about

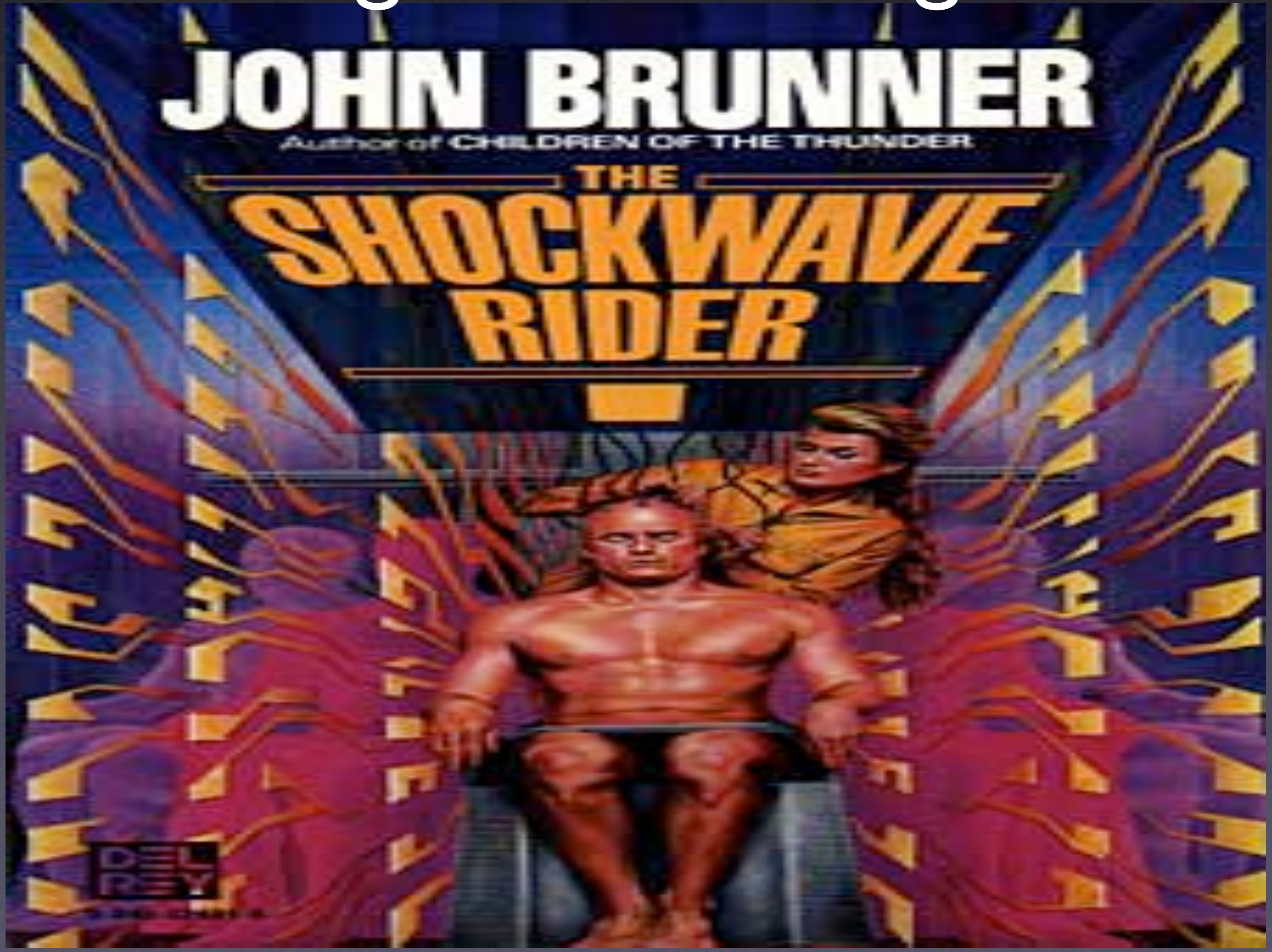
- MULTI MACHINE PROGRAMS (!): Programs which can copy (replicate) themselves from one computer to others
- Experiences
  - How they did it (model, implementation)
  - The problems encountered
  - What they learned
- Feasibility experiment and prototyping
  - **What was wrong with the world?**
    - A worm had not been demonstrated before, and published about (ditto for a distributed computation)
    - **They had a concept, needed to learn more: so they did it.**
- MADE POSSIBLE BY THE COMMODITY DEVELOPMENT of computers (microprocessor, RAM, disk, network)

Inspired by "...father and mother of all tapeworms..."

- 1975: The Shockwave Rider by John Brunner"
- "Tapeworm" program running loose through the network
- Breeds by itself
- Moves to places
- Immortal
- Consume resources

Inspired by "...father and mother of all tapeworms..."

- 1975: The Shockwave Rider by John Brunner"
- "Tapeworm" program running loose through the network
- Breeds by itself
- Moves to places
- Immortal
- Consume resources



Side track

- Page 172, third column: "..once called *distributed* **computing**. Unfortunately, that particular phrase has already been co-opted by those who market fairly ordinary terminal systems; thus, we prefer to characterize these as *programs which span machine boundaries* or *distributed* **computations**"

The Blob

- Science Fiction Movie, 1950s: Lifeform growing by absorbing others
- Computational model based on this idea :)
  - expand when cycles become available
    - need migration of code and data
  - retreat when users start using their workstations
    - need migration of state and results
  - Also called VAMPIRE PROGRAMS
    - hiding at day, fly by night
- Check out a 1980's project: EMERALD (U. of Washington)

Prologue

- John Shoch and Jon Hupp @ Xerox Palo Alto Research Center, 1982
- Homogeneous Computing Environment
- 100 Alto computers personal use, also see
- Set of servers: file servers, printer-servers, boot-servers
- Ethernet Local Area Network
- No (TCP/)IP, they used PUP (Xerox Parc's "IP")
- Programs written in BCPL for Alto.
- **Bravo** WYSIWYG text editor, **Laurel** Electronic mail program, **Press** Document printing program, and Games
- Single user, no user level multi-programming
- **"Idle computer"==running memory**

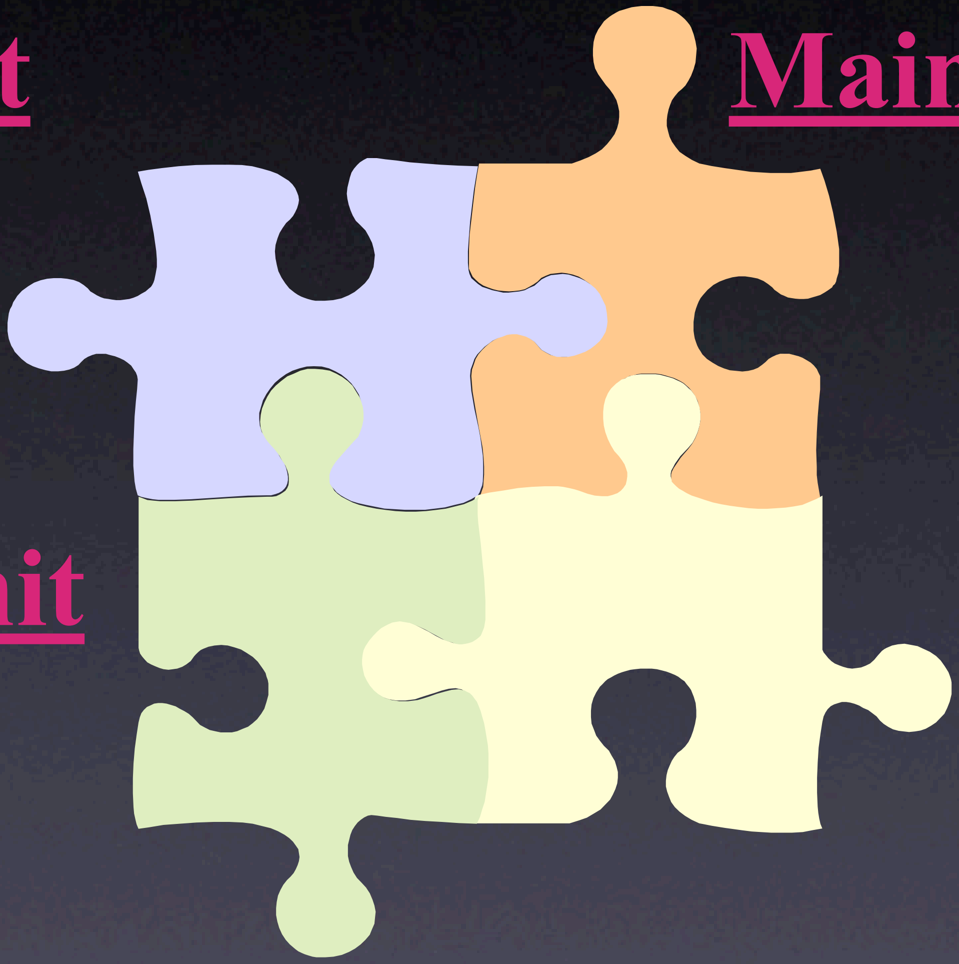
Alto Computer

- http://en.wikipedia.org/wiki/Xerox_Alto
- Butler Lampson
- Xerox Parc 1973
- First computer to use
  - desktop metaphor
  - GUI
- Not commercial, a few thousand used

Worm

- Motivation:
  - To use idle machines (resources) effectively
- Definition:
  - “A computation that lives on one or more machines. Programs on individual machines are segments of a worm.”
- A worm program initiates on one machine, reach out and find unused or idle machines, and grow to encompass these resources
- Vampire programs

Worm



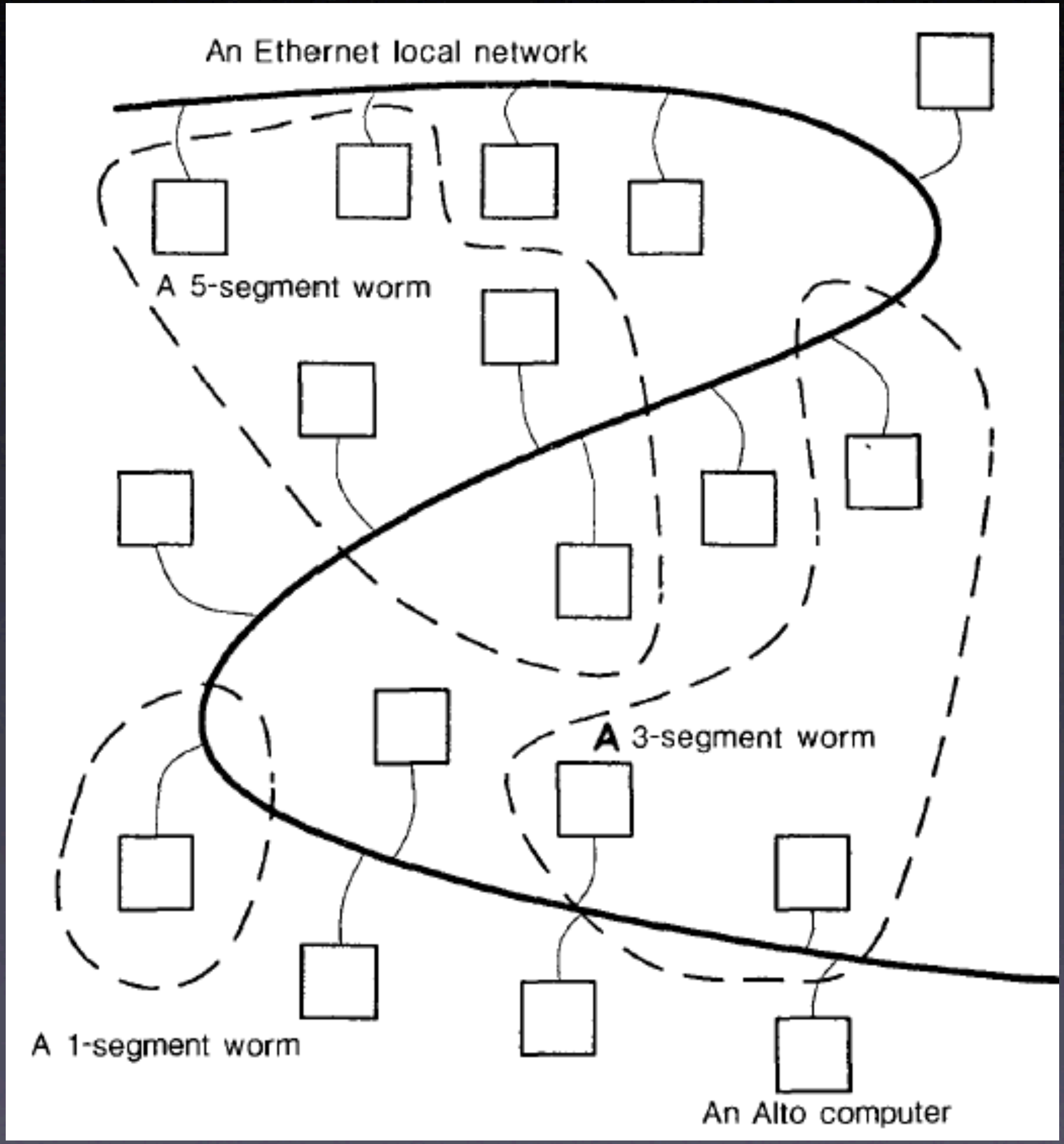
Worm Init

Main program

Segment Init

Data

Schematic of Worm Programs



- Simplest case: Each segment carries a number indicating total segments (== machines) in a worm
- If a segment dies, others find an idle machine, initialize and add it to the worm
  - => need comm. between segments "all the time"
  - => worm moves
- Worm is a mechanism (infrastructure) used to maintain segments
- User programs built on top of this mechanism

Right :)

- Do not use the *local* disk (file servers OK)
  - because the computer may not *have* a disk
  - because it is *antisocial* to use others disks
- Users must have *confidence* in that the worm will behave
  - “we have been able to assure users that there is not even a disk driver included within any of the worm programs””

Mechanisms

- Starting a worm segment
    - Initialization code at the start
- “Must grow!”: Locating idle machines
    - Simple Q: “Are you free?”
        - *Memdiagnostics* answers *YEPP*
    - *But which addresses should each worm SEGMENT use?*
        - ***repeat*** *try own++ until YEPP then copy segment to new computer* ***break loop*** *%got you, new host has a segment. It will try to spread, I will not*

How to get the segment copied to new host - and running

- Can not remotely take control over an Alto and start a program, let alone force it to restart it over the network
  - an Alto must **voluntarily** reboot from local or remote disk
- Booting an idle machine
  - ***Remote*** segment **asks** new host, please, to **boot** through network from a **given** location
    - the boot code is now the segment itself!
      - The Alto now runs the Worm segment
        - single program + OS code

Mechanisms

- **Intra-worm communication** (remember: they need to know if a segment dies)
    - Ethernet supported multicast would have been nice - they did not have this - and it would not have worked for a WAN anyhow
    - Magic to achieve multi-cast support
        - Pseudo-multicast: **each** worm is given a **unique** physical host address to use. All “participants” (computers) rename themselves to this address
            - NB: worms can not share computers, must partition the computers amongst the worms, coordination is a drag :)
        - Brute force multicast: Periodically all segments send and get state from the others
            - n(n-1) “packets” per update
                - no problem, many small worms (3-6 will secure survivability :))
        - If a segment dies: will be discovered, the rest agrees on whom of them will locate a new machine and copy itself to it

Segment finished

- Releasing a machine: Upon finishing, segment invokes network boot procedure to reload memory diagnostic programs
- What if machine crash while running the segment or during reboot?
  - well, then it has crashed :)
    - no support for a reboot initiated from network
    - must walk up to it and cycle power

Segment to Segment

- Segments communicate to keep the worm alive, maintaining the set number of segments
- Alive?
  - Heartbeat
  - Ping
- Agree on which segment should try to invade a new host if the number of segments are too low (or kill if too many)

One Big Worm vs. Cooperating Smaller Worms

- Almost as an aside they mention "cooperating smaller worms", this was in 1982
- CSP - communicating sequential processes, Hoare, 1978
- But think about the implications: s2s vs. w2w interaction
    - P2P - peer to peer

Oops

No, Mr. Sullivan, we can't stop it! There's never been a worm with that tough a head or that long a tail! It's building itself, don't you understand? Already it's passed a billion bits and it's still growing. It's the exact inverse of a phage--whatever it takes in, it adds to itself instead of wiping... Yes, sir! I'm quite aware that a worm of that type is theoretically impossible!

But the fact stands, he's done it; and now it's so goddamn comprehensive that it can't be killed. Not short of demolishing the net!

--John Brunner, The Shockwave Rider

Out-of-control Worms

- Challenge: Controlling worm growth while maintaining stable behaviour
  - Copy of program corrupted during migration: after a segment was downloaded, it would not behave (initialize) correctly, and crash => machine dead until cycle power. Somewhere the Worm concluded: I *need* to get up to my correct number of segments => all machines will eventually crash
    - The authors had designed in a way to kill all segments: Inject a special packet into network: HALT, it said to all segments.
      - NB: of course, all segments had to have code to read and react to this packet. COULD have been corrupted as well of course...
  - High strung worm: very sensitive, no attempts to be robust: panics easily.
  - Low strung worms: Worms can die out because the segments are too relaxed in copying themselves to new computers
  - Unstable worm: grows rapidly due to lack of coherence between segments (inconsistent state, partion of network)
- Real challenge: Unlike viruses and trojans, worms have caused greater havoc on Internet
  - November 2, 1988: Morris worm
  - July 19, 2001: Code-Red (CRv2)
    - 359,000 computers infected in 14 hours
    - $2.6 billion

Controlling the worms

- Exchange of more information
- Use of checks and error detection
- Self-destruction of segments thinking they themselves are the problem
  - hmm, then the worm can die by suicide
- Worm watcher: keep an eye on the worms, keep them down or halt, record state

Applications

- **No app specific comm. between segments**
  - Existential worm: Stay alive, print(Greetings from Worm Segment i)
  - Billboard worm: Displays an image on all machines with a segment
- **App. specific comm between segments**
  - Alarm clock worm: A program on some Alto provided a user interface: set alarm. It then contacted a segment of the Alarm worm with the new alarm info. This segment propagated its state to all other segments. Alarm clock will survive machine crashes! CONSISTENCY ISSUES.
    - how would the user program find a segment? (use a port read by all segments, listen on the Ethernet, or more modern approaches)
- **Centralized control by Master node**
  - Multimachine animation using a worm: segments used to produce frames, master displays
    - master poll slaves-slaves return a frame-master send new objects to compute to segments
    - can use one larger worm or several smaller (HIERARCHY)
  - Diagnostic worm for the Ethernet

Concepts still being investigated

- self-replication
- migration
- distributed coordination
- control
- defenses (against malicious programs)

Epilogue

- One of the earliest experiments in distributed computing and process migration
- Valuable experiences in
  - Distributed computing
  - Moving from machine to machine
- Experiments and experiences in Worms have been quite useful to develop support of process migration in later systems
  - Systems to use idle workstations or distribute load in networked environment. E.g. Butler, NEST.
- Control needed because problems will happen