Network Algorithmics, Introduction

George Varghese

January 13, 2005

Introduction

- What is Internet Algorithmics?
- Warm up Exercise

Will assume some knowledge of basic network protocols but hope to teach you the rest (including some working knowledge of hardware).

What is Internet Algorithmics?

- **Definition:** Network Algorithmics is the use of an interdisciplinary systems approach, seasoned with algorithmic thinking, to address network processing bottlenecks at servers, routers, and other networking devices.
- For introductory purposes, divide into 2 parts: the problems (network bottlenecks) and the techniques (via a warm-up example).

Part 1, The Problems

Network Bottlenecks

- How to reconcile ease of use and performance in networking.
- Ease of use via network abstractions (e.g., socket interfaces, prefix-based forwarding).
- Wthout care such abstractions exact a large performance penalty when compared to the raw fiber capacity.
- Network algorithmics seeks to address performance gap. We examine two fundamental categories of networking devices, *endnodes* and *routers*.

Endnode Algorithmics

- *Computation versus Communication:* Endnodes are about general purpose computing, unknown and varied computational demands.
- *Vertical versus Horizontal Integration:* Many companies supplying subsystems. kernels designed to tolerate unknown and potentially buggy applications.
- *Complexity of computation:* Endnode protocol functions are more complex (application, transport) compared to routers.

4 Artifacts of Endnode Software

- *1. Structure:* Complexity and vastness leads to structured, modular code. API firewall between kernel and unknown application.
- *2. Protection:* need to protect applications from each other, and protect the operating system from applications.
- *3. Generality:* Core routines such as buffer allocators and the scheduler are written with the most general use in mind.
- *4. Scalability in concurrent connections:* Simple data structures that work well for a few concurrent connections become major bottlenecks in a server environment.

Endnode Algorithmics Preview

Bottleneck	**Chapter**	**Cause**	**Sample Solution**
Copying	*5*	*Protection, structure*	*Copying many data blocks without OS Intervention (e.g., RDMA)*
Context Switching	*6*	*Complex Scheduling*	*User-level protocol implementations* *Event driven web servers*
System calls	*6*	*Protection, structure*	*Direct channels from applications to Drivers (e.g., VIA)*
Demuxing	*7*	*Scaling with # of endpoints*	*BPF and PathFinder*
Timers	*8*	*Scaling with # of timers*	*Timing wheels*
Checksums/ CRCs	*9*	*Generality* *Scaling with link speeds*	*Multibit computation*
Protocol code	*9*	*Generality*	*Header Prediction*

Router Algorithmics

- "Router" used generically for other network devices such as bridges, switches, gateways, monitors, and security appliances, and for protocols other than IP such as FiberChannel.
- Typically, pecial-purpose devices devoted to networking. Little structural overhead within a "router", lightweight operating system.

Forces that affect Router Performance

- *Scale:* Scaling in bandwidth and population.
- *Services:* Very success of the Internet boom requires guarantees in terms of performance, security, and reliability.

Preview of Router Algorithmics

Bottleneck	Chapter	Cause	Sample Solution
Exact Lookups	10	Link speed scaling	parallel hashing
Prefix Lookups	11	link speed scaling Prefix database size scaling	Compressed multibit tries
Packet Classification	12	Service differentiation Link speed and size scaling	Decision tree algorithms Hardware parallelism (CAMs)
Switching	13	Optical-electronic speed gap Head of line blocking	Crossbar switches Virtual Output Queues
Fair Queueing	14	Service differentiation Link speed scaling Memory scaling	Weighted fair queuing Deficit Round Robin DiffServ, Core Stateless
Internal Bandwidth	15	Scaling of internal bus speeds	Reliable striping
Measurement	16	Link speed scaling	Juniper's DCU
Security	17	Scaling in number and intensity of attacks	Traceback with Bloom Filters Extracting worm signatures

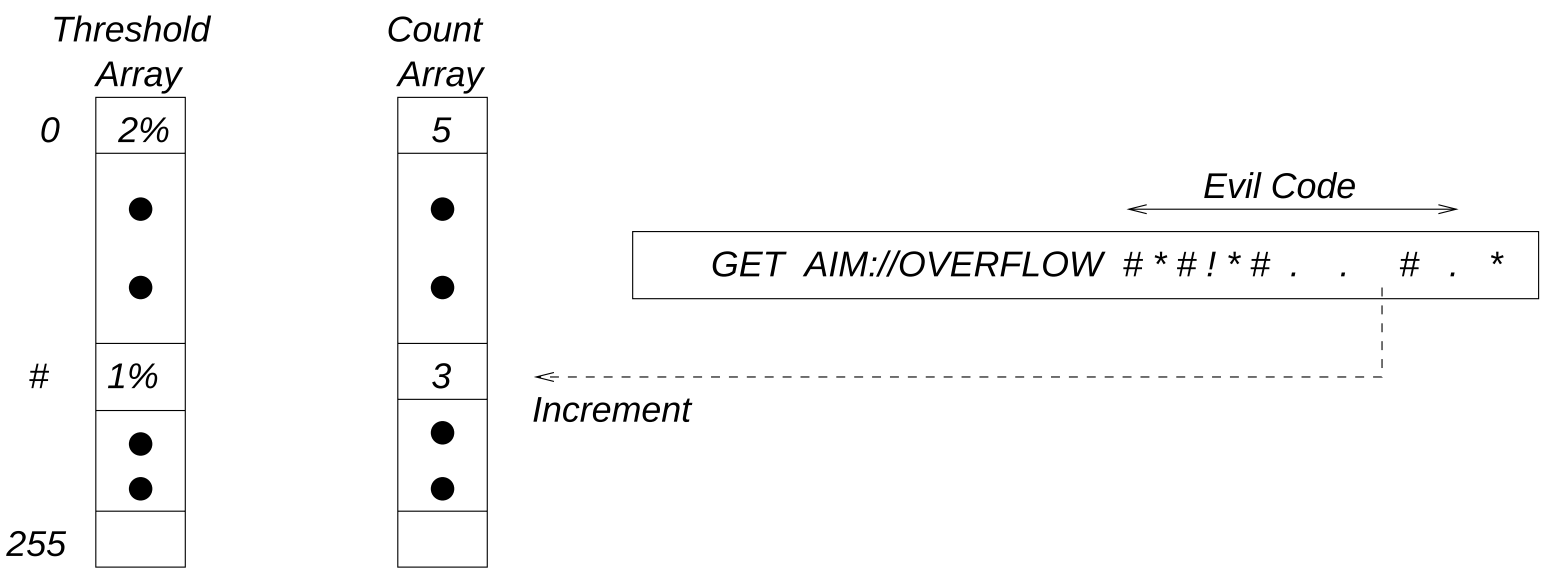
Part 2, The solutions

- Besides problems, also a way of thinking.
- Can (and will) talk about in abstractions but fastest way is to sample via a warm-up example. So, here goes . . .

Warm Up Exercise: Scenting Buffer Overflow

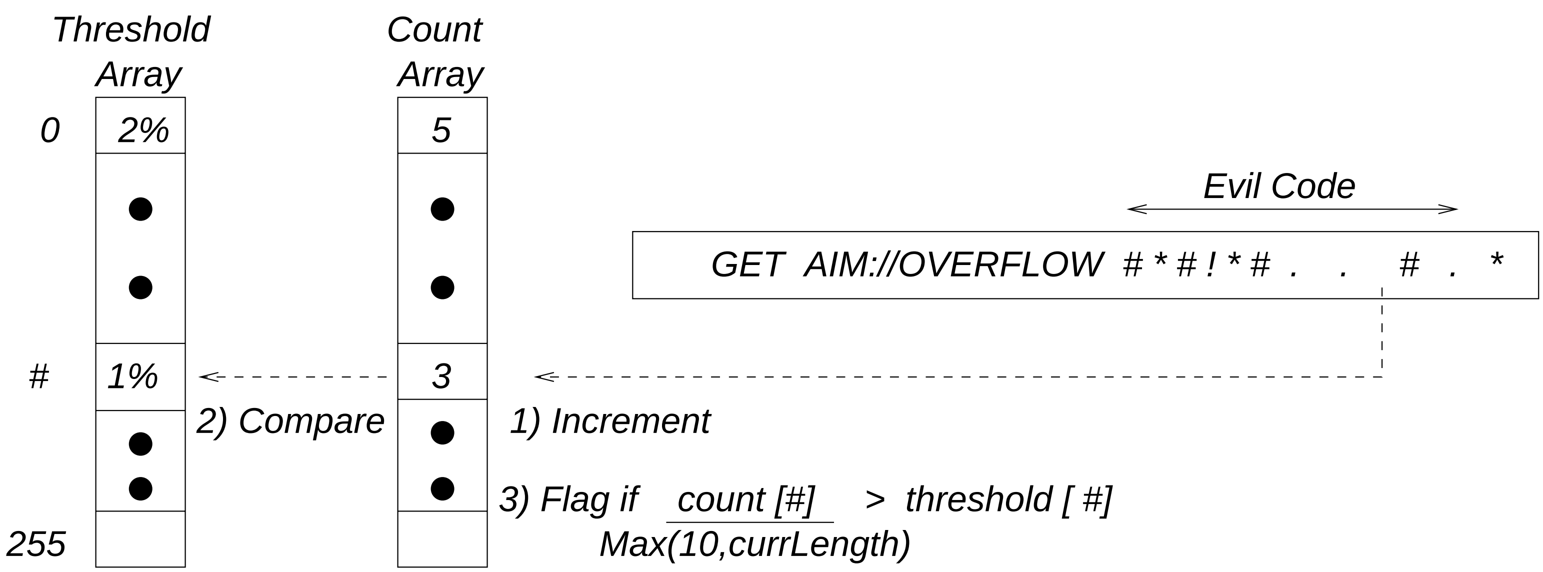
- **Observation:** Large frequency of uncommon characters within code buried in say URL.
- **Problem:** Can we flag such packets for further observation? Problem due to Mike Fisk of LANL/UCSD.

Solution Stage 1: Strawman



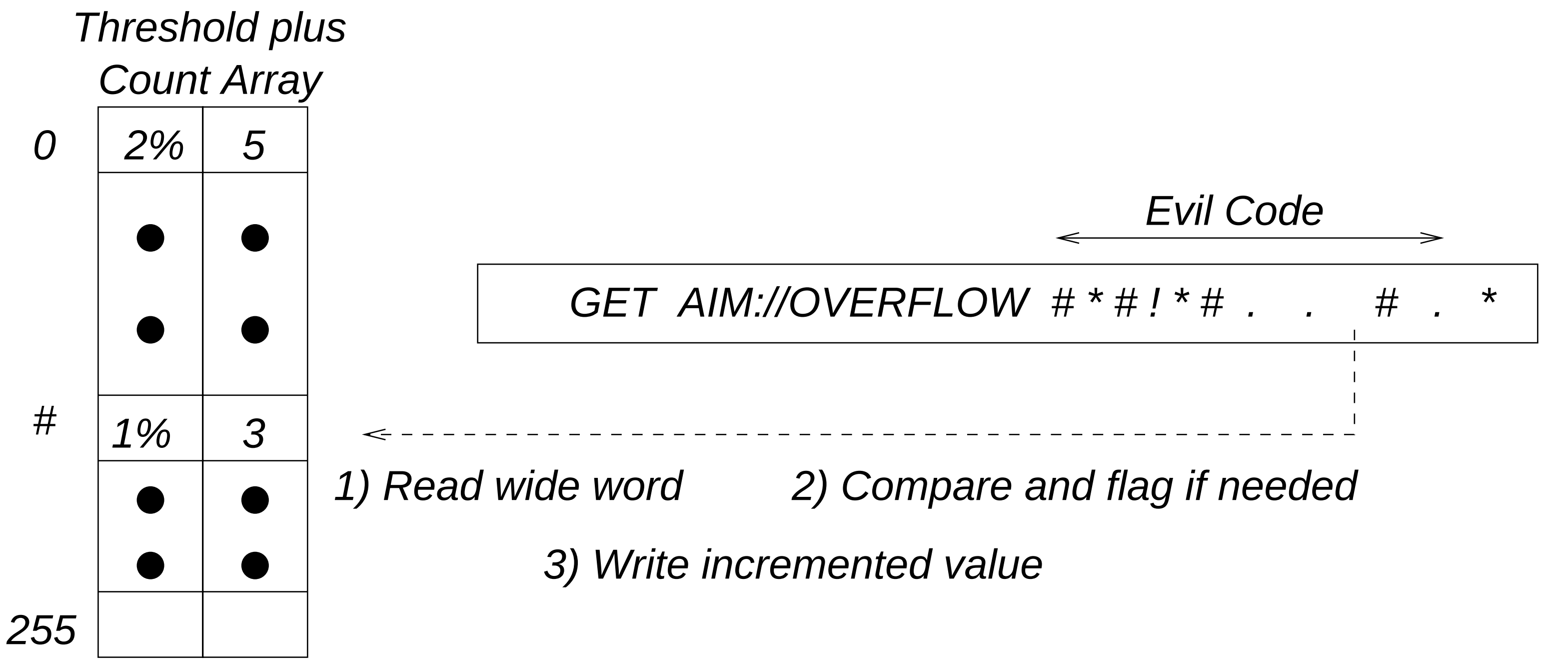
- **Strawman:** Increment each character count as packet enters. At end, flag if any count over threshold.
- **Problem:** Slow. Adds a second pass over large array with 256 Reads to memory after packet is processed.

Stage 2: Oliver asks for Less



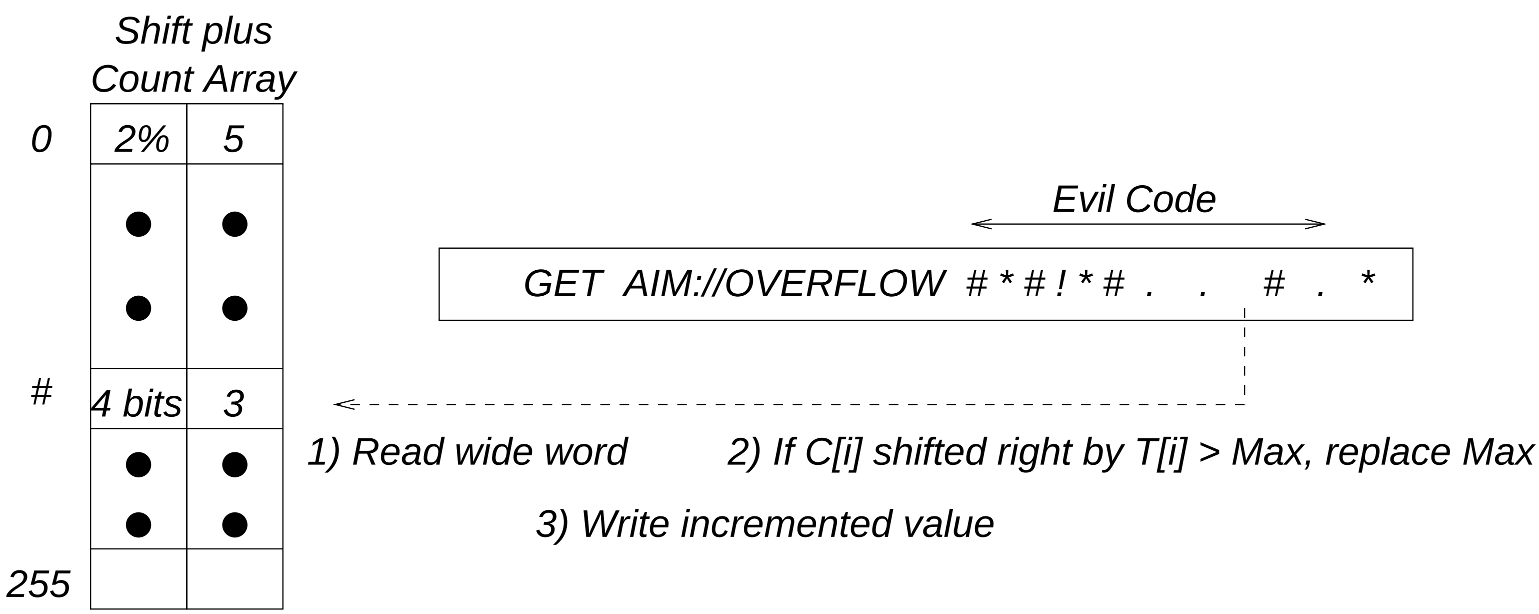
- **Idea:** "Please Mr. Architect can we relax specification so that we flag if any count is over threshold wrt *current* and not total length.
- **Problem:** Still two reads and 1 write to memory per byte.

Stage 3: Coalesce arrays



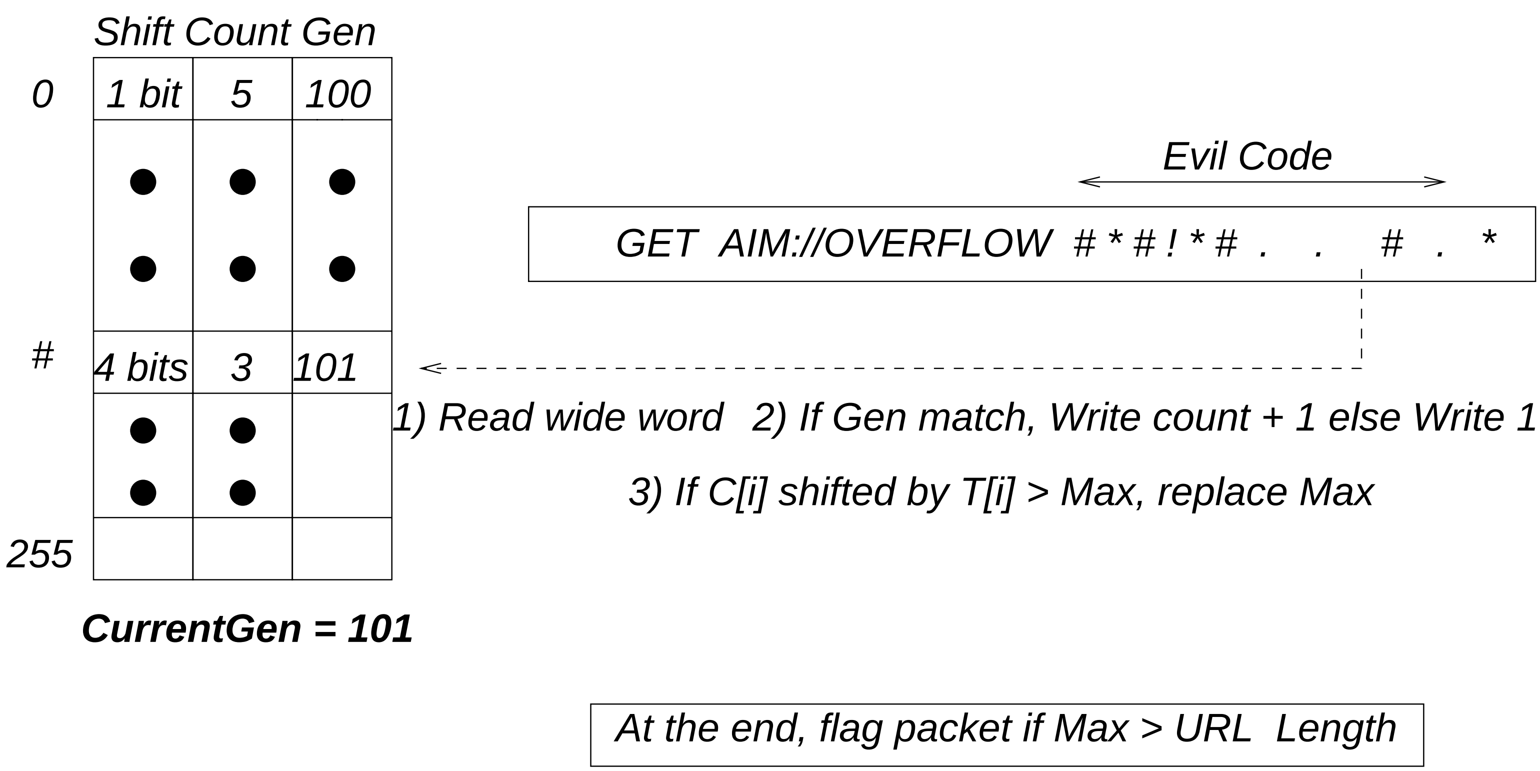
- **Observation:** Can use wide memories and coalesce two arrays to reduce to 1 Read + 1 Write.
- **Problem:** Hard/slow to do divisions in hardware.

Stage 4: Approximate!



- **Observation:** Have false positives anyway. Round up thresholds to powers of 2, replace division by shift.
- **Problem:** How to initialize counts.

Stage 5: Lazily initialize



- **Observation:** If we keep a generation number, can lazily initialize counts as long as we guarantee initialization of all older generations before number wraps around.

Network Algorithmics

Attacking Internet bottlenecks via:

- Better hardware: At Gigabits and Terabits, we would be hard pressed not to understand basics of busses, switches, memories, and chips. *wide word access, logic to manipulate a word.*

- Better system decompositions: Relax system specifications, shift functions in space (other components) or time (system instantiation times). *e.g., Approximating thresholds as powers of 2, lazy initialization*

- Better algorithms: Clever ways of solving isolated functions though with metrics and models that sometimes differ from standard algorithms. *None in this example, none in many implementations, but some in this tutorial!.*

. . . So a reasonable plan of attack is

We will continue our study of Part I of Network algorithmics as follows:

- **Chapter 2, Models:** Hardware and architecture for implementing patterns
- **Chapter 3, Principles:** Efficient design principles.
- **Chapter 4, Problems:** Simple problems to try out principles

Part 2 of the book will cover endhost bottlenecks (where OS issues are crucial) and Part 3 will cover router bottlenecks (where hardware issues are paramount). Part 4 of the book will cover newer problems in security and measurement.