CS4450

Computer Networks:
Architecture and Protocols

The LAST one.
Where's the puck going?

Rachit Agarwal

Announcements

- **Final exam**
    - 12/05, in-class
    - **Not** unlimited time

- **Review session**
    - Tomorrow, 1PM, zoom (see Ed Discussions)

- **Material: everything covered in class**

What Have We Done so far in reliable transport?

- Started from first principles
    - Correctness condition for reliable transport
- … to understanding **why feedback from receiver is necessary** (sol-v1)
- … to understanding **why timers may be needed** (sol-v2)
- … to understanding **why window-based design may be needed** (sol-v3)
- … to understanding **why cumulative ACKs may be a good idea**
    - Very close to modern TCP

What Have We Done so far in reliable transport?

- **To understanding TCP-specific mechanisms**
  - Connections
  - Segments, sequence numbers, ACKs
  - **Retransmissions** (based on timeout, and duplicate ACKs)
  - Flow control
  - **Congestion Control**
    - cwnd increase (no congestion)
    - cwnd decrease (congestion, isolated & extreme)

- **To understanding TCP properties**
  - **Sawtooth behavior**
  - Convergence under stable state
  - **Max-min fair resource allocation**

The Many Failings of TCP Congestion Control

1. Fills up queues (large queueing delays)
2. Every segment not ACKed is a loss (non-congestion related losses)
3. Produces irregular saw-tooth behavior
4. Biased against long RTTs (unfair)
5. Not designed for short flows
6. Easy to cheat

(1) TCP Fills Up Queues

- TCP only slows down when queues fill up
    - High queueing delays
- Means that it is not optimized for latency
    - What is it optimized for then?
        - **Answer: Fairness (discussion in next few slides)**
- And many packets are dropped when buffer fills
- Alternative 1: Use small buffers
    - Is this a good idea?
    - Answer: No, bursty traffic will lead to reduced utilization
- Alternative: **Random Early Drop (RED)**
    - Drop packets on purpose **before** queue is full
    - A very clever idea, but results in unfairness

(2) Non-Congestion-Related Losses?

- If packets are corrupted (no congestion)
  - TCP would think the network is congested
  - Incorrect response!

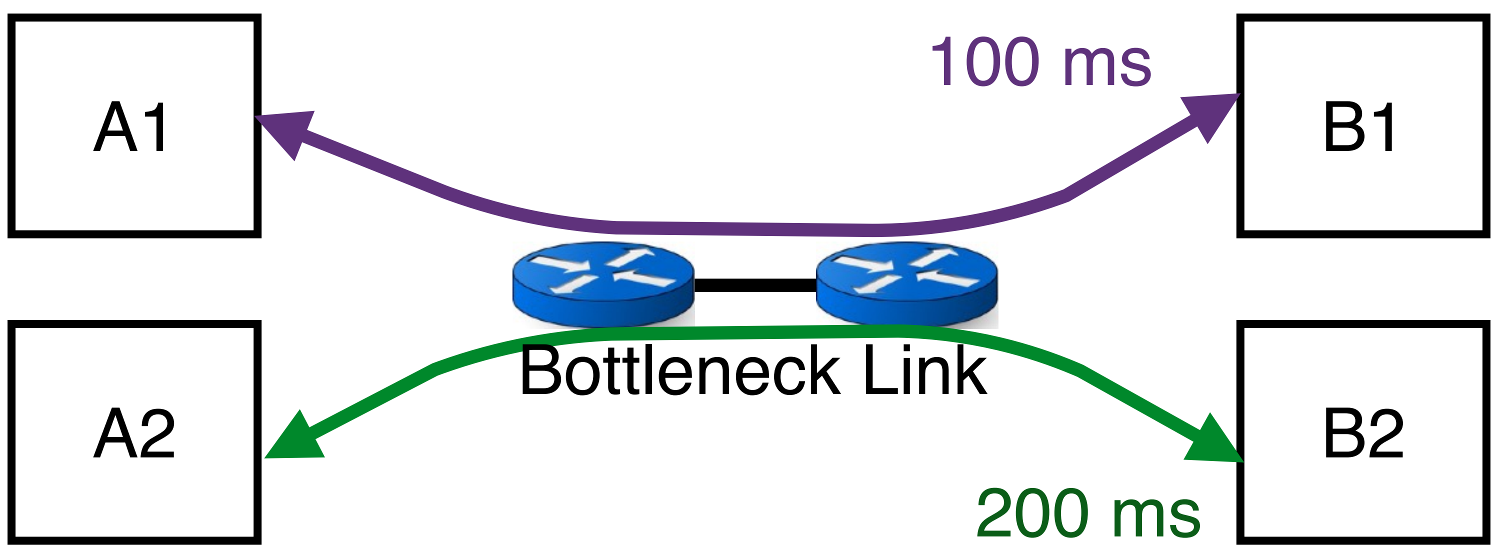
- Several possible solutions:
  - Can use **Explicit Congestion Notification (ECN)**
  - As routers get congested, they mark the packet with ECN
  - Thus, receiver can differentiate between corruption & congestion

(3) Sawtooth Behavior Uneven

- TCP throughput is “choppy"
  - Repeated swings between W/2 to W
- Some apps would prefer sending at a steady rate
  - E.g., streaming apps
- A solution: “Equation-based congestion control”
  - Ditch TCP’s increase/decrease rules and just follow the equation:
  - **[Matthew Mathis, 1997] TCP Throughput = MSS/RTT sqrt(3/2p)**
    - **Where p is drop rate**
  - Measure drop percentage p and set rate accordingly
- Following the TCP equation ensures we’re TCP friendly
  - I.e., use no more than TCP does in similar setting

(4) Bias Against Long RTTs

- Flows get throughput inversely proportional to RTT
- **TCP unfair in the face of heterogeneous RTTs!**
- [Matthew Mathis, 1997] TCP Throughput = MSS/RTT sqrt(3/2p)
    - Where p is drop rate
- Flows with long RTT will achieve lower throughput



(5) How Short Flows Fare?

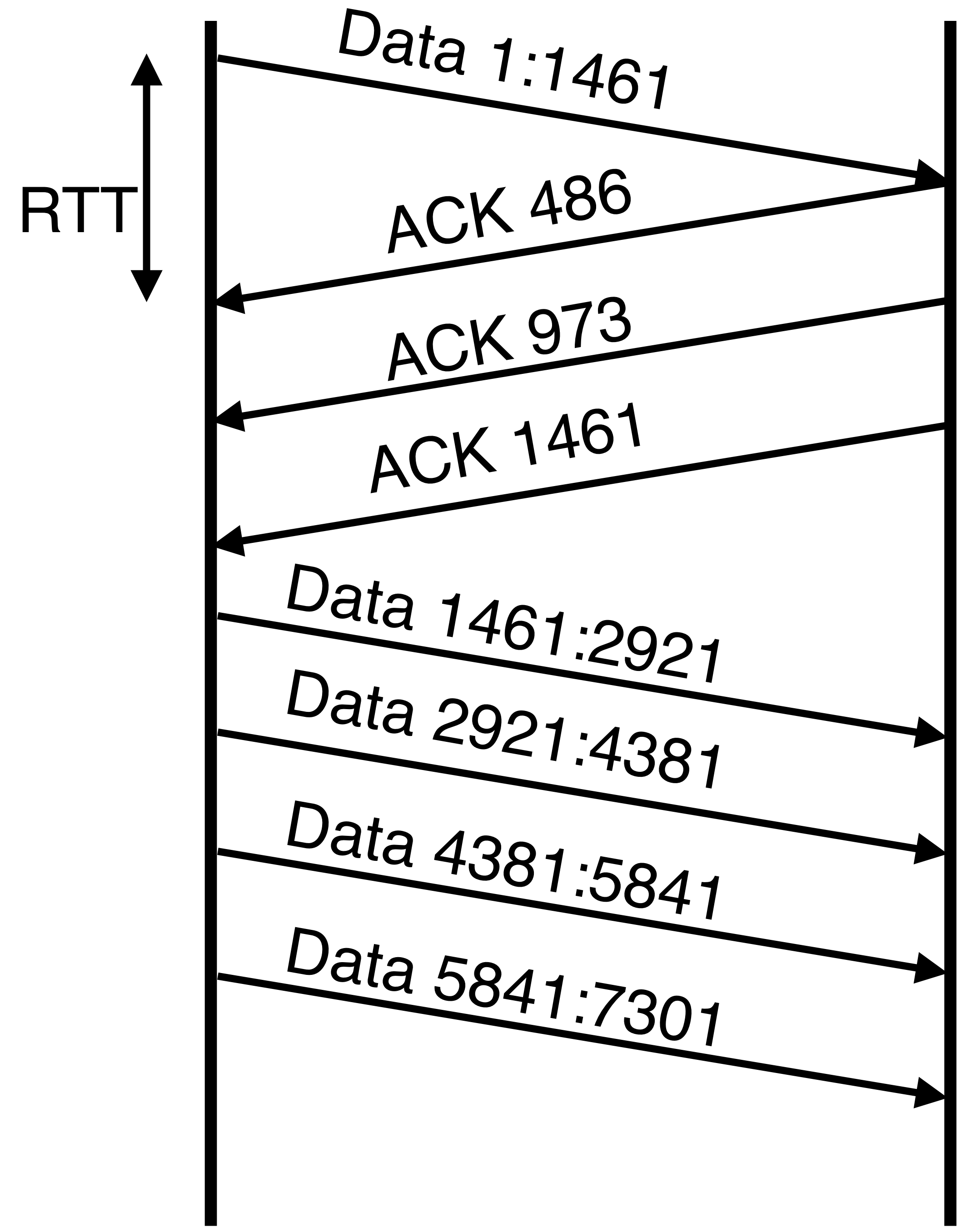
- Internet traffic:
  - Elephant and mice flows
  - Elephant flows carry most bytes (>95%), but are very few (<5%)
  - Mice flows carry very few bytes, but most flows are mice
    - 50% of flows have < 1500B to send (1 MTU);
    - 80% of flows have < 100KB to send
- Problem with TCP?
  - Mice flows do not have enough packets for duplicate ACKs!!
  - Drop ~=~ Timeout (unnecessary high latency)
  - These are precisely the flows for which latency matters!!!
- Another problem:
  - Starting with small window size leads to high latency

(6) Cheating

- TCP was designed assuming a cooperative world
- No attempt was made to prevent cheating
- Many ways to cheat, will present three

Cheating #1: ACK-splitting (receiver)

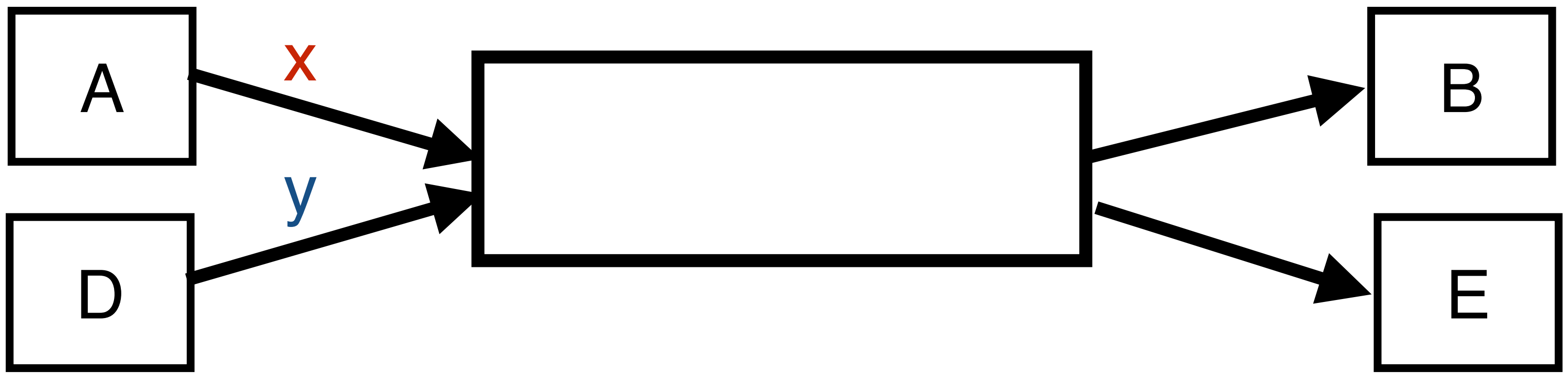
- TCP Rule: grow window by one MSS for each valid ACK received
- Send **M** (distinct) ACKs for one MSS
- Growth factor proportional to **M**



Cheating #2: Increasing CWND Faster (source)

- TCP Rule: increase window by one MSS for each valid ACK received
- Increase window by **M** per ACK
- Growth factor proportional to **M**

Cheating #3: Open Many Connections (source/receiver)



- Assume
  - A start 10 connections to B
  - D starts 1 connection to E
  - Each connection gets about the same throughput
- Then A gets 10 times more throughput than D

Cheating

- Either sender or receiver can independently cheat!
- **Why hasn’t Internet suffered congestion collapse yet?**
    - Individuals don’t hack TCP (not worth it)
    - Companies need to avoid TCP wars
- How can we prevent cheating
    - Verify TCP implementations
    - Controlling end points is hopeless
- Nobody cares, really

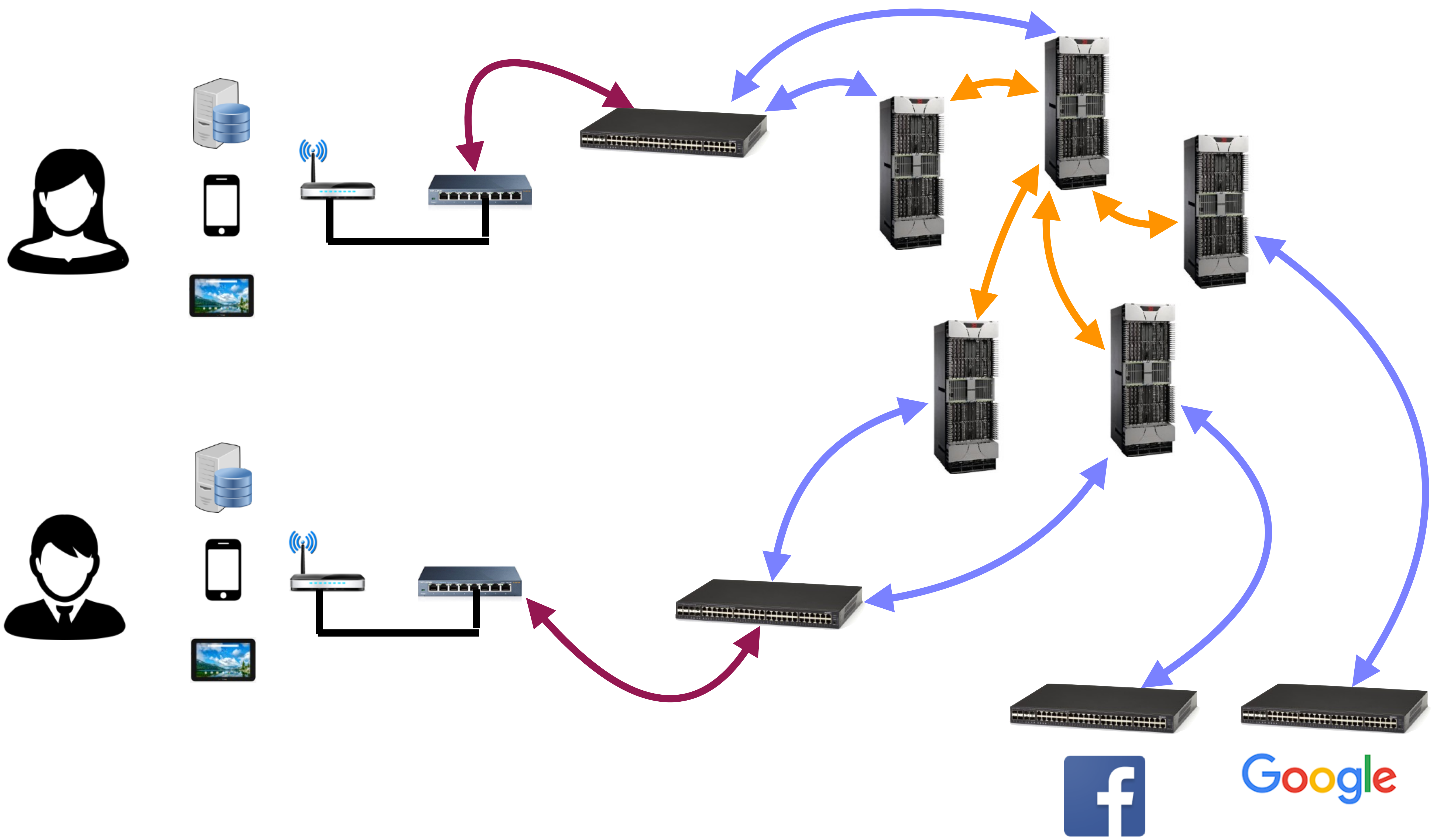
Now you know about computer networking

as much as I do :-)

Taking 25 steps back!

What is a computer network?

A set of network elements connected together, that implement a set of protocols for the purpose of sharing resources at the end hosts



Sharing networks

- **Two approaches**
  - Reservation (circuit switching)
  - Statistical multiplexing (packet switching)
- **Motivation for WHY modern networks use "packets"**
- **How to implement this?**

The end-to-end story

- Application opens a **socket** that allows it to connect to the **network stack**
- Maps **name** of the web site to its **address** using **DNS**
- The network stack at the source embeds the address and **port** for both the source and the destination in **packet header**
- Each **router** constructs a **routing table** using a distributed algorithm
- Each router uses destination address in the packet header to look up the **outgoing link** in the routing table
    - And when the link is free, forwards the packet
- When a packet arrives the destination:
    - The network stack at the destination uses the port to forward the packet to the right application

Realizing end-to-end design: Three Principles

- How to break system into modules
  - **Layering**
- Where are modules implemented
  - **End-to-End Principle**
- Where is state stored?
  - **Fate-Sharing**

Five Layers (Top - Down)

- **Application:** Providing network support for apps
- **Transport (L4):** (Reliable) end-to-end delivery
- **Network (L3):** Global best-effort delivery
- **Datalink (L2):** Local best-effort delivery
- **Physical (L1):** Bits on wire

Link Layer (L2)

- **Broadcast medium:** Ethernet and CSMA/CD
- **We studied that Broadcast Ethernet does not scale to large networks**
    - Motivation for switched Ethernet
- **Broadcast storm:** if using broadcast on switched Ethernet
    - Motivation for Spanning Tree Protocol
- **Limitations of Spanning Tree Protocol:**
    - Low bandwidth utilization, high latency, unnecessary processing
    - Does not scale to the entire Internet
    - Motivation for **routing protocols** in the Internet

Network Layer (L3)

- **Internet Protocol:**
  - Addressing, packet header as an interface, routing
- **Routing tables:**
  - Correctness and validity: Dead ends, loops
  - A collection of spanning trees, one per destination
- **Constructing valid routing tables (within an ISP)**
  - Link-state and distance-vector protocols
  - Focused a lot on learning via examples
  - Can still have loops: failures remain to be a pain
- **How to use routing tables**
  - **Packet header as an interface**
  - Learnt why packet headers look like the way they do

Network Layer (L3), Cont.

- **Internet Protocol:**
  - Addressing, packet header as an interface, routing
- **Addressing:**
  - Link layer uses “flat” addresses
  - **Does not scale to Internet**: motivation for IP addresses
  - **Scalability challenges**: Routing table sizes, #updates
  - Solution: **Hierarchical addressing**
- **Forwarding**
  - **Switch architecture**
  - Longest Prefix matching for forwarding at line rate
  - Scheduling using priorities

Network Layer (L3), Cont.

- **Internet Protocol:**
  - Addressing, packet header as an interface, routing
- **Limitations of link-state and distance-vector routing:**
  - Require visibility of the entire Internet
  - **ISPs do not like that:** motivation for Inter-domain routing
  - **Border Gateway Protocol**
    - A simple modification of distance-vector protocol
- **Routing with policies**
  - **Customer-provider-peer relationships**
  - Gao-Rexford policies
- **Completes the network layer: provides connectivity**

Details for complete picture

- **DHCP: Dynamic Host Configuration Protocol**
  - For each host to figure out its IP address, local DNS, first-hop router
- **ARP: Address Resolution Protocol**
  - For finding other servers on the same local area network (L2)
  - Mapping from IP addresses to names (MAC addresses)
- **Domain Name System**
  - **Mapping Human readable destination names to IP addresses**
  - Hierarchical structure

Transport Layer

- **Goals of reliable transport**
  - **Correctness condition**
  - Why do we need ACKs, timers, window-based design

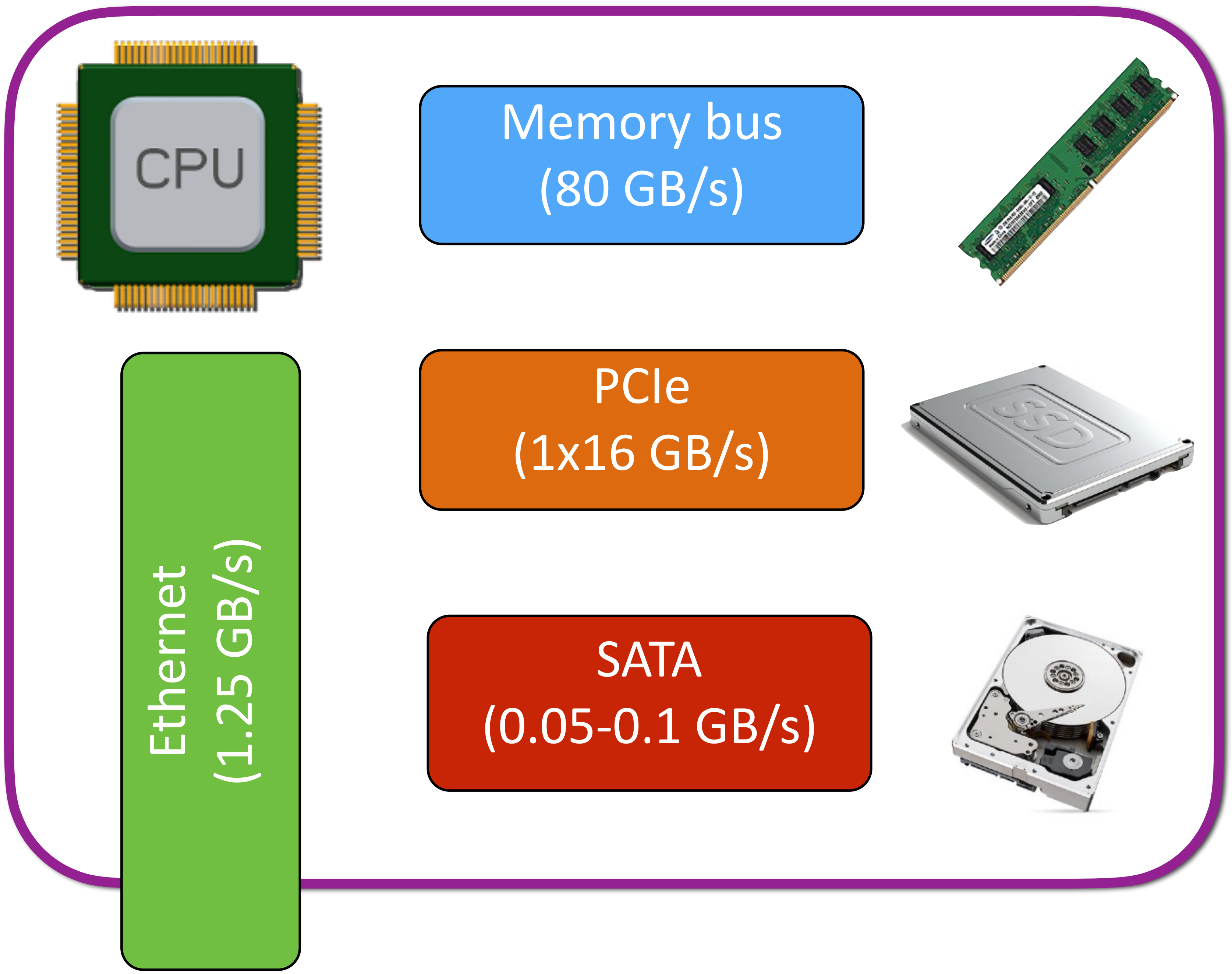
- **One realization of reliable transport: TCP**
  - Mostly implementation details following the above design
  - For **max-min fairness**, flow performance and utilization
  - **Flow control**
    - Ensuring the sender does not overwhelm the **receiver**
    - Via receiver advertised window size
  - **Congestion control**
    - Ensuring the sender does not overwhelm the **network**
    - Slow start, Additive-increase Multiplicative-decrease, timeouts

Taking 1 step forward!

Skate where the puck’s going, not where it’s been!

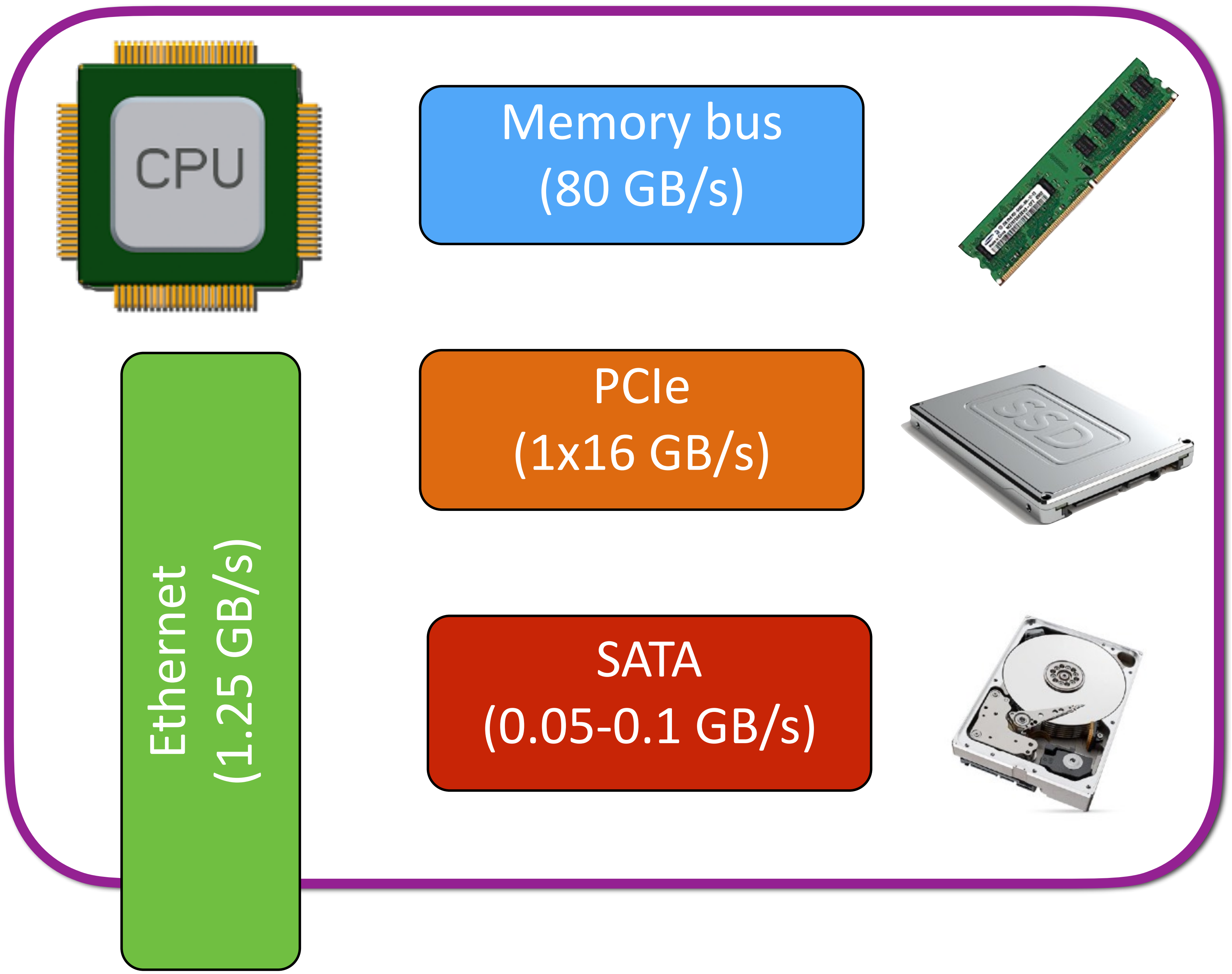
- Walter Gretzky

Where is the puck right now?

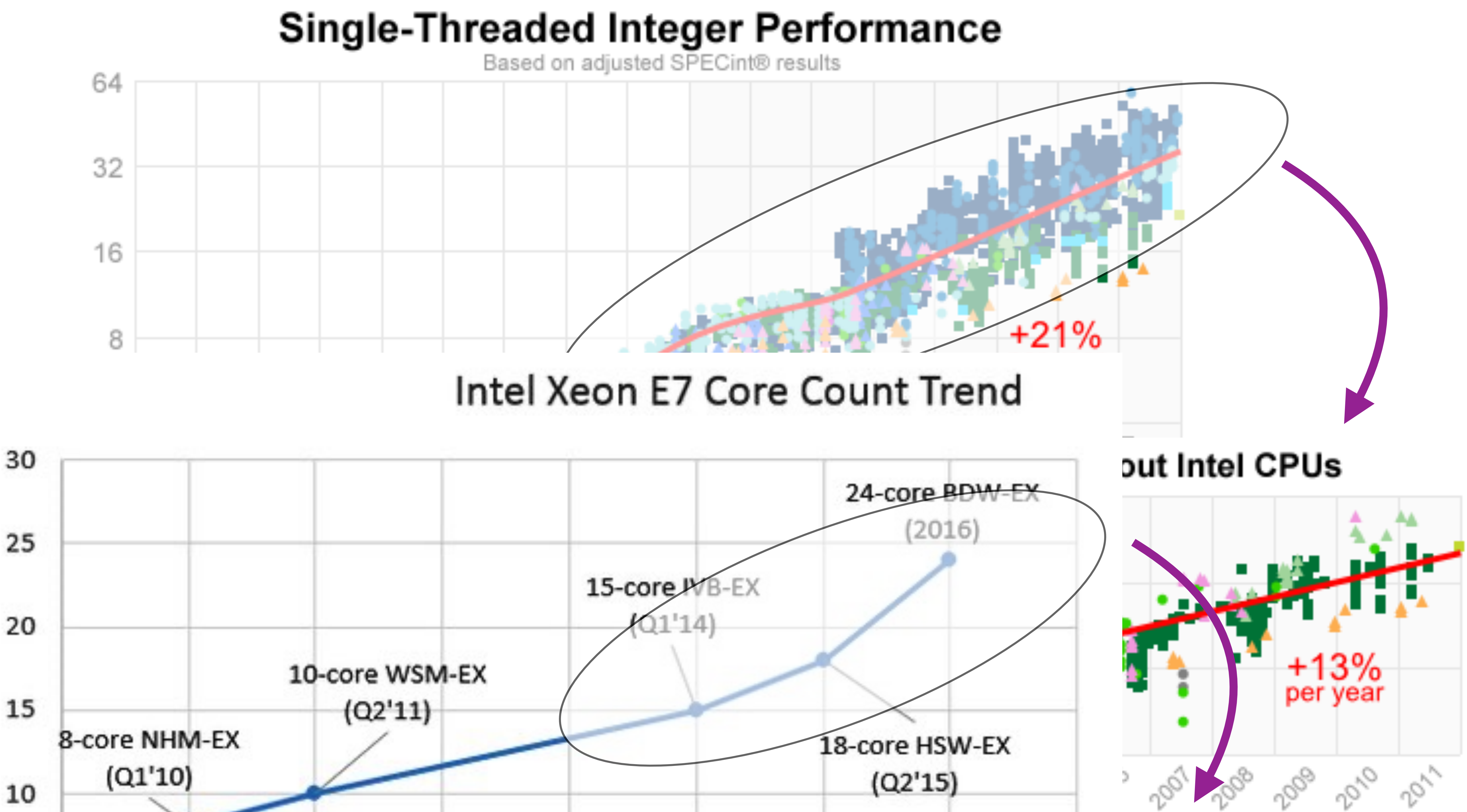


Size (TB)	Random Access (us)	Seq. Access (GB/s)
0.1	0.1	80
1	25	1x
10	4000	0.1x

Where is the puck going?



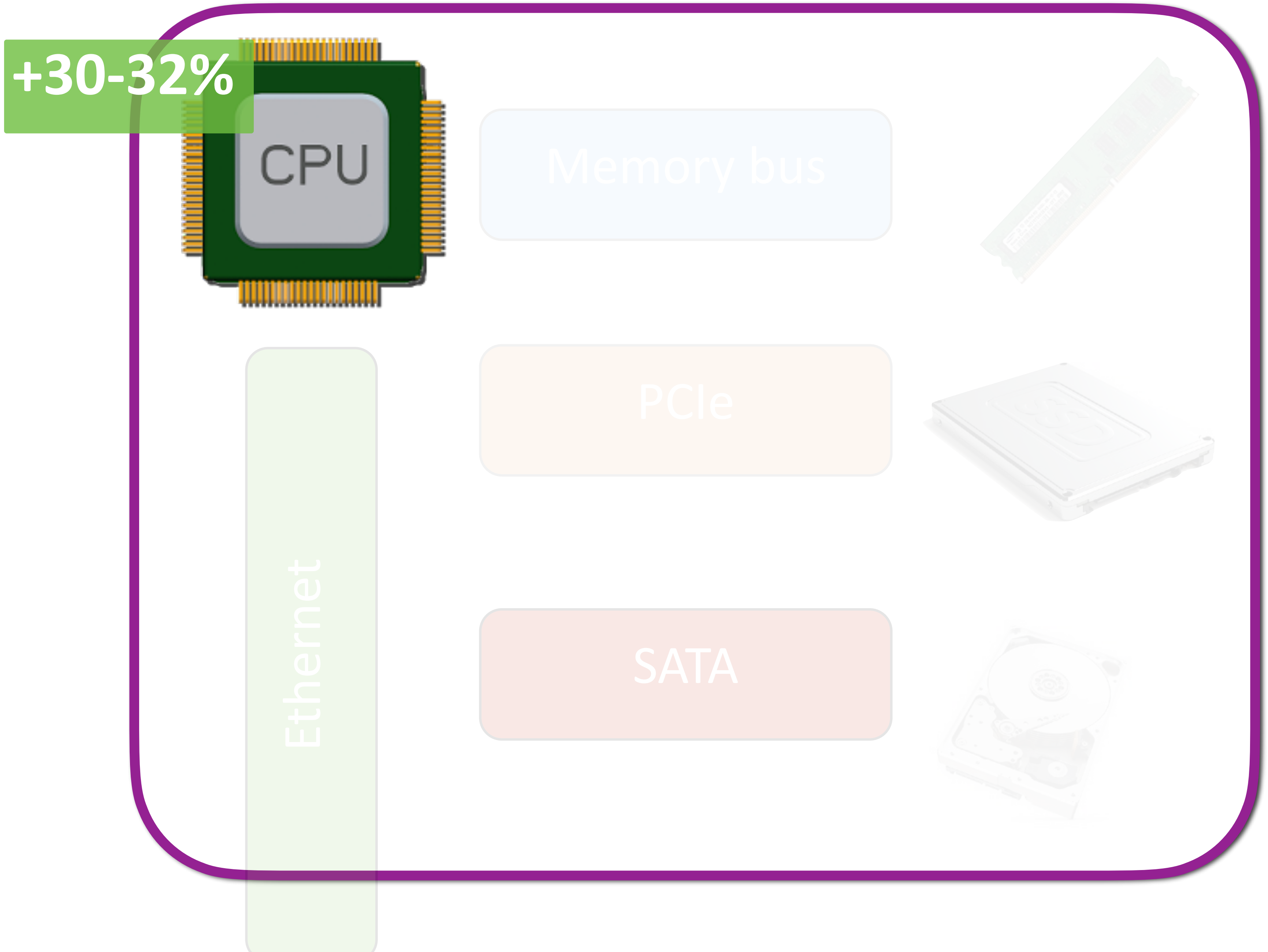
Where is the puck going? (CPU performance)



2016: +18-20%

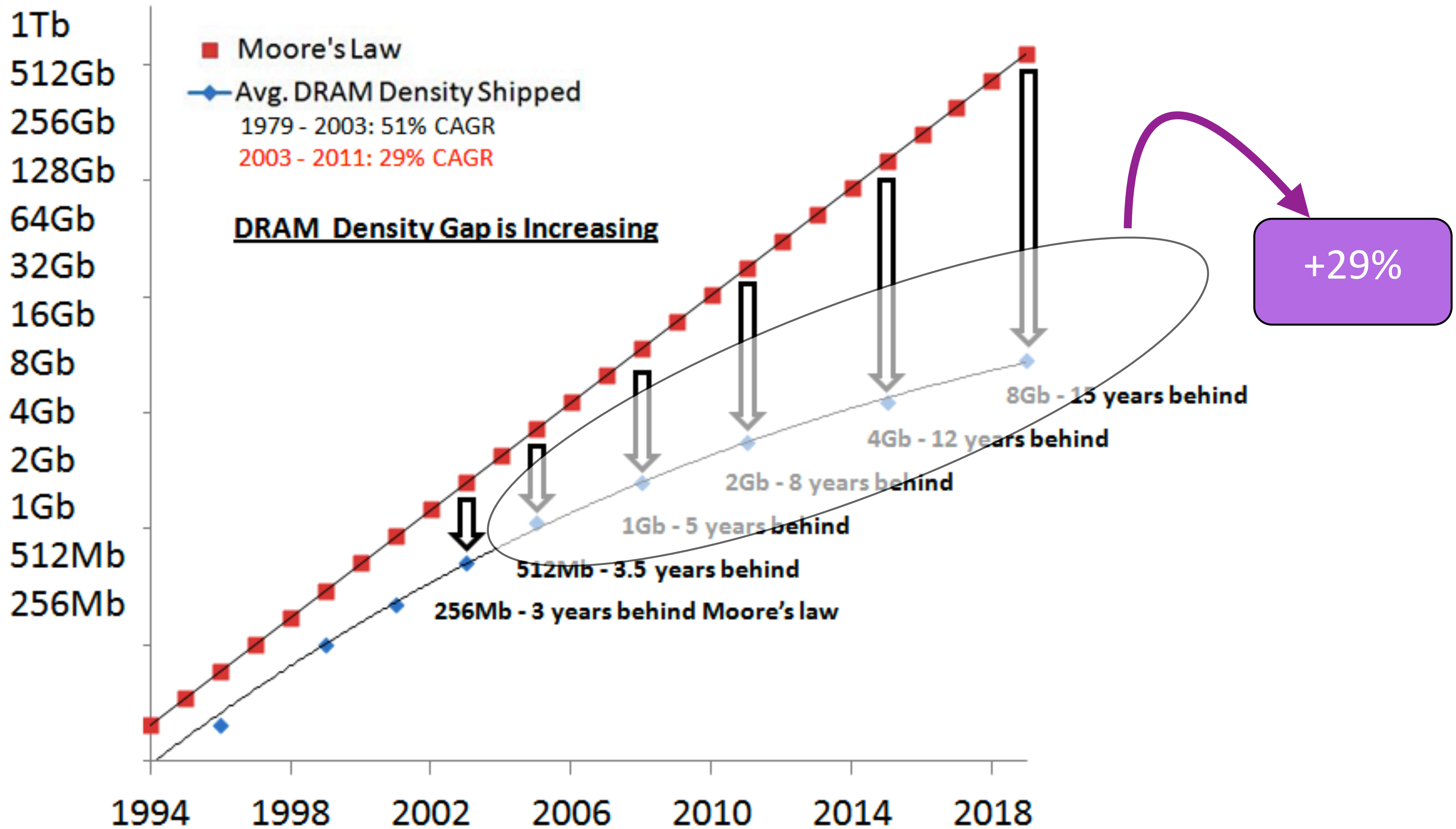
2016: +10%

Where is the puck going?

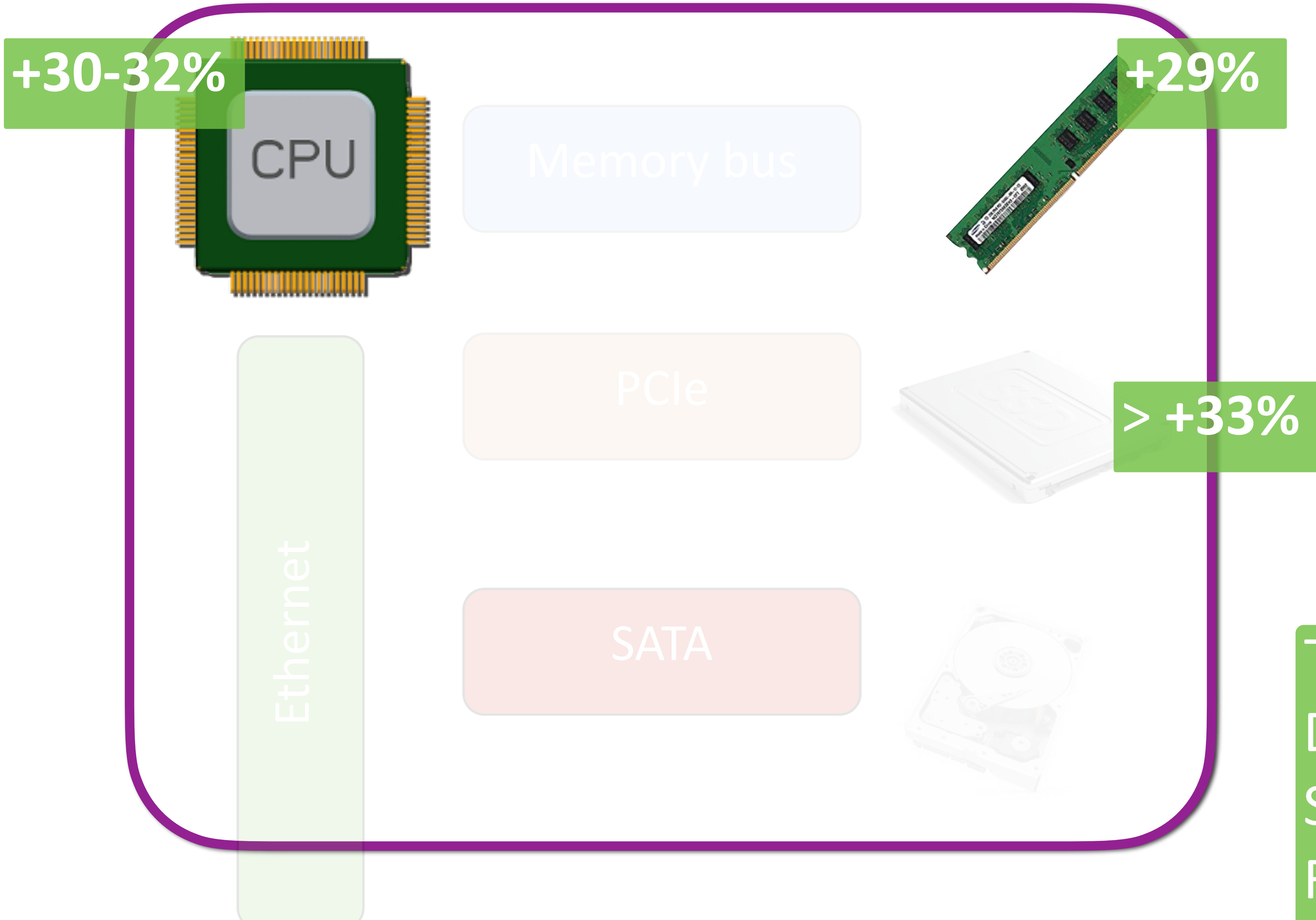


- **#Cores: +**18-20%
- **Per core: +10%**

Where is the puck going? (DRAM capacity)



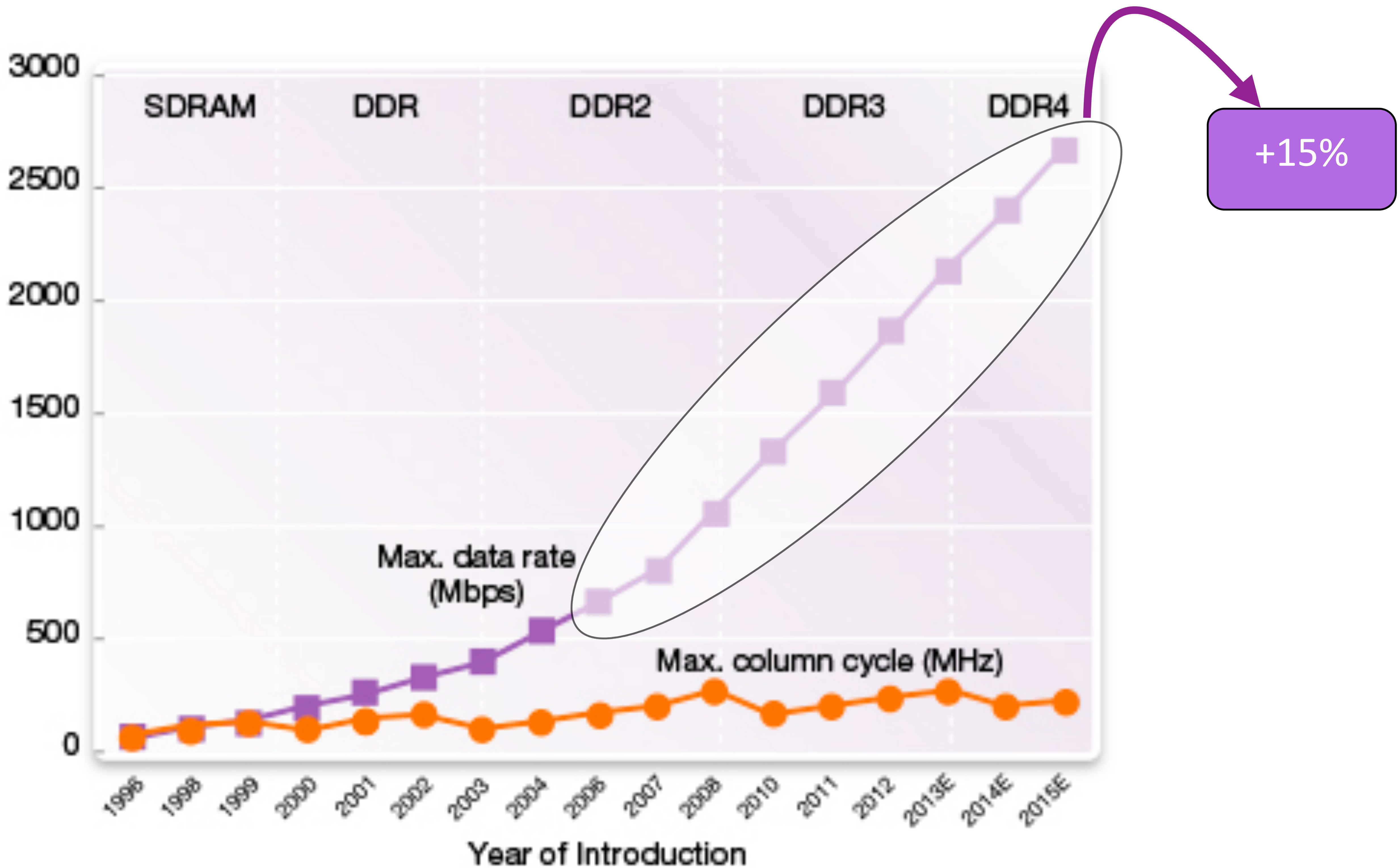
Where is the puck going?



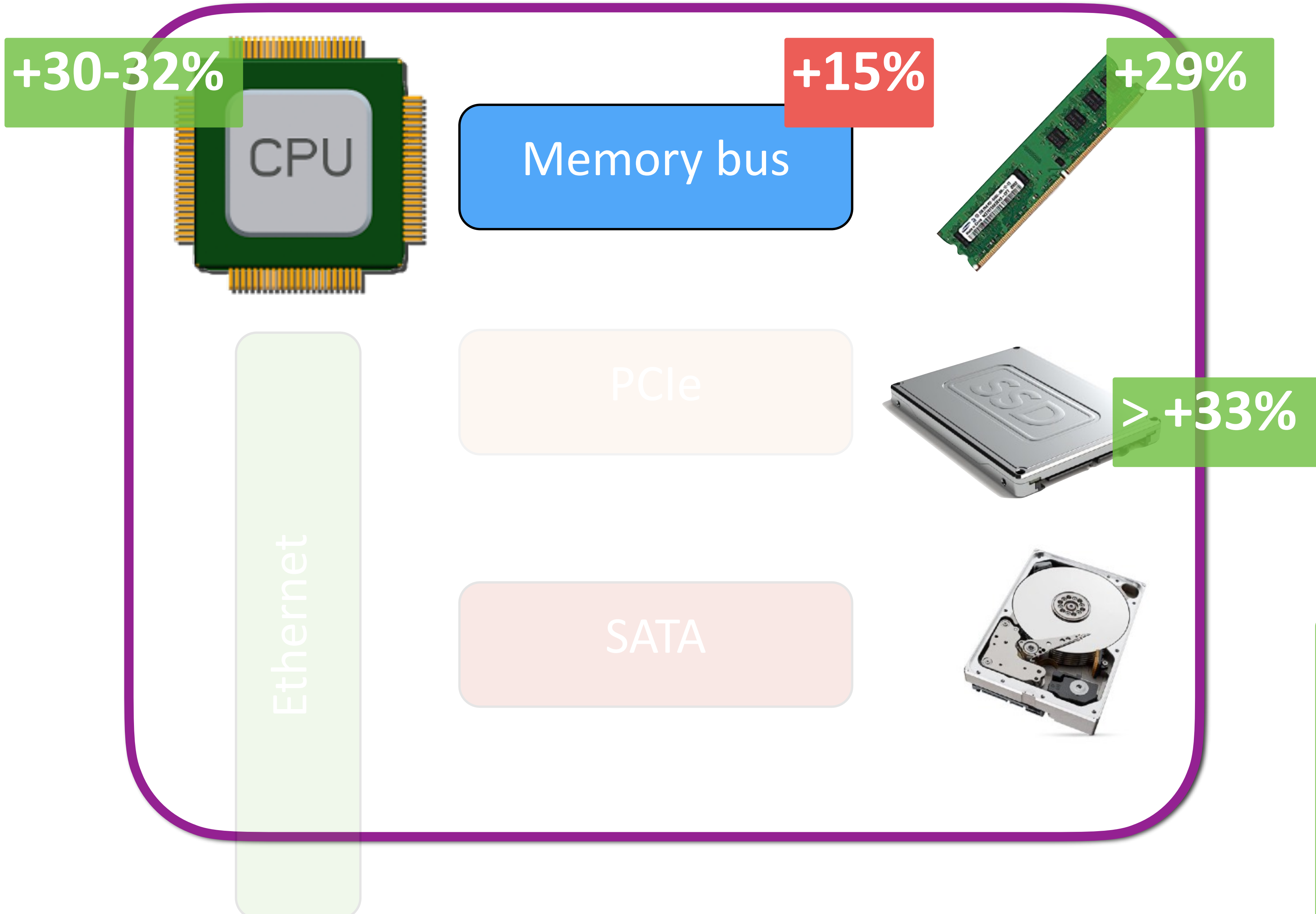
Tape is dead,
Disk is tape,
SSD is disk,
RAM is the king!

- Jim Gray

Where is the puck going? (Memory bus)



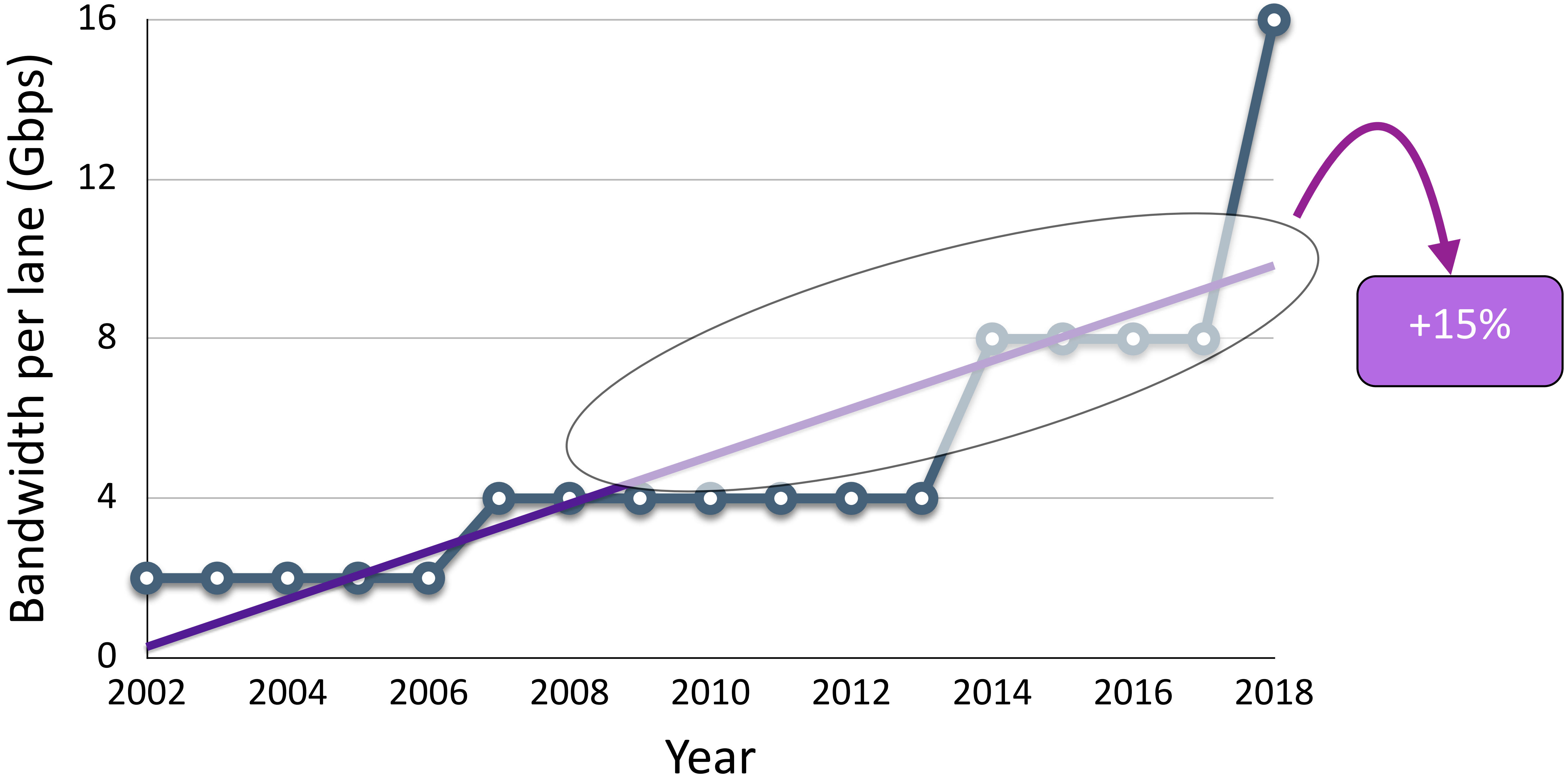
Where is the puck going?



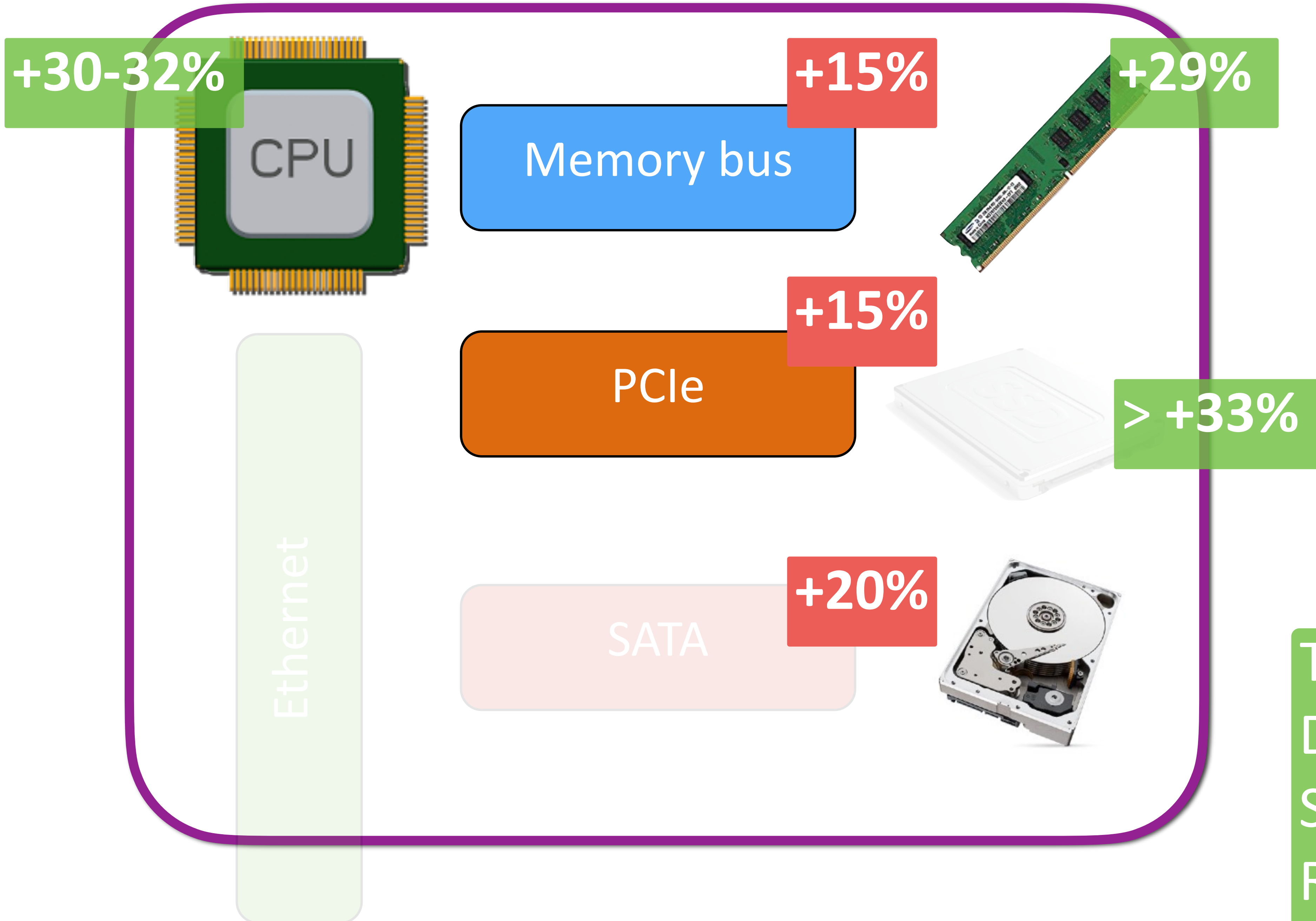
Tape is dead,
Disk is tape,
SSD is disk,
RAM is the king!

- Jim Gray

Where is the puck going? (PCIe)



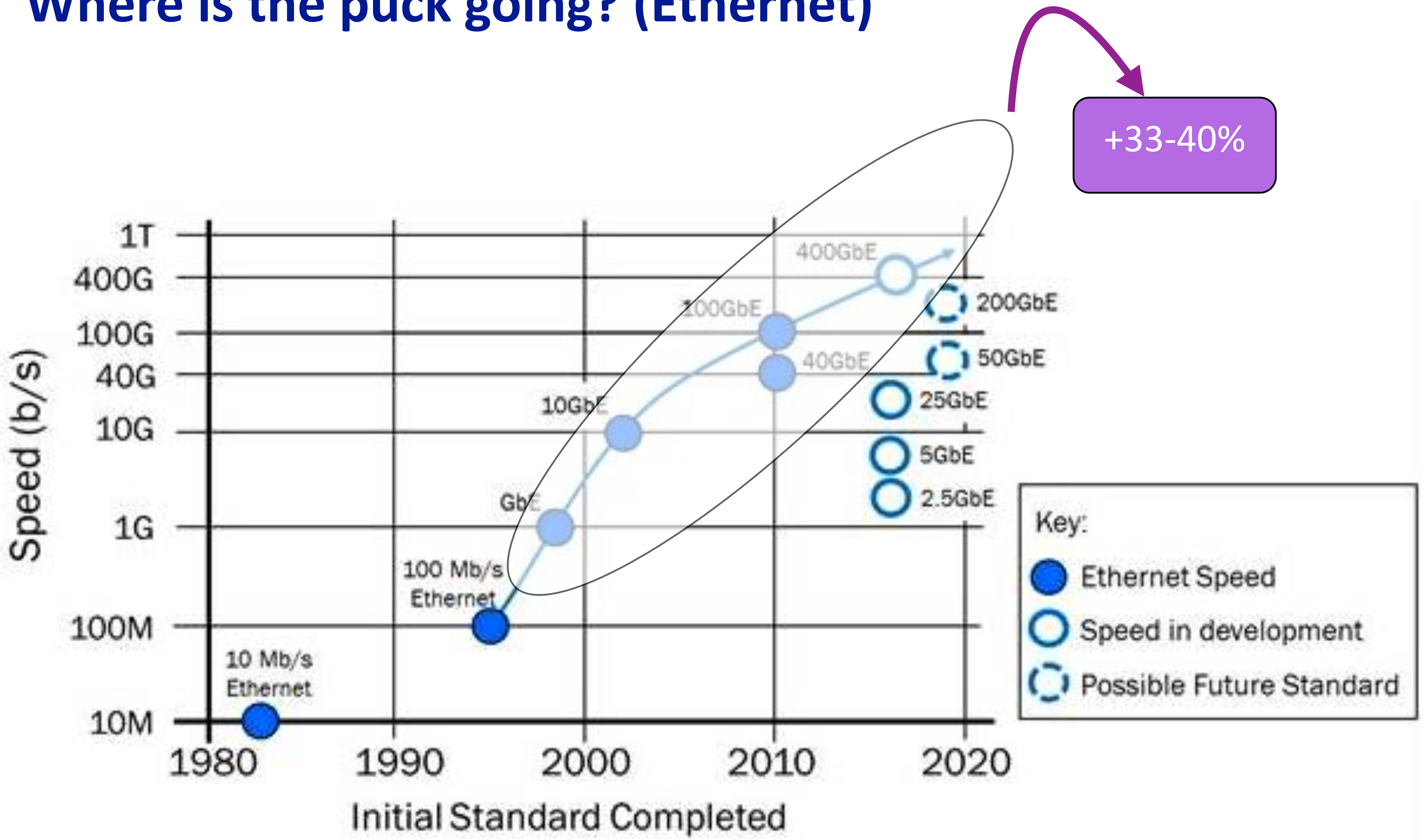
Where is the puck going?



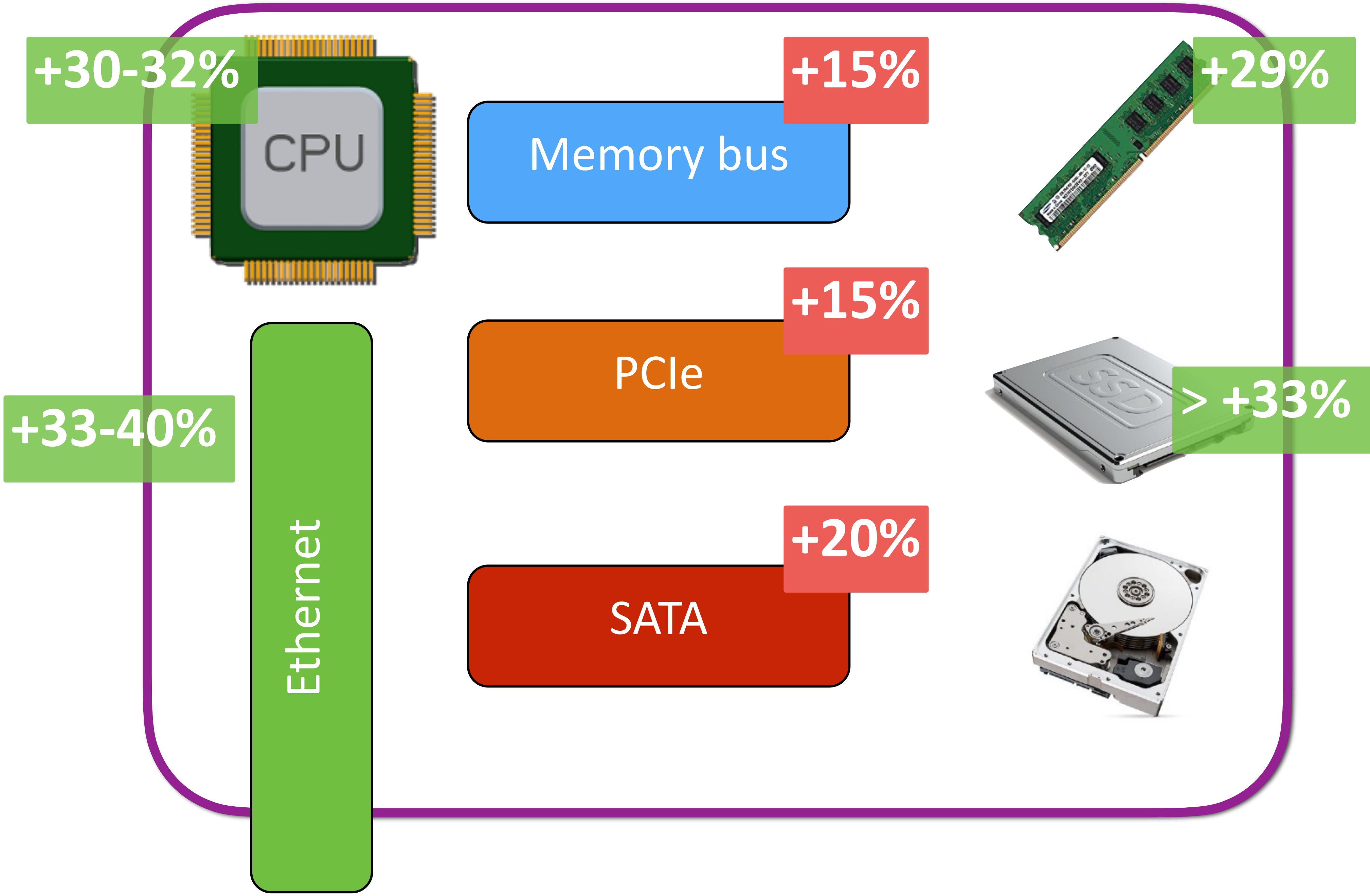
Tape is dead,
Disk is tape,
SSD is disk,
RAM is the king!

- Jim Gray

Where is the puck going? (Ethernet)



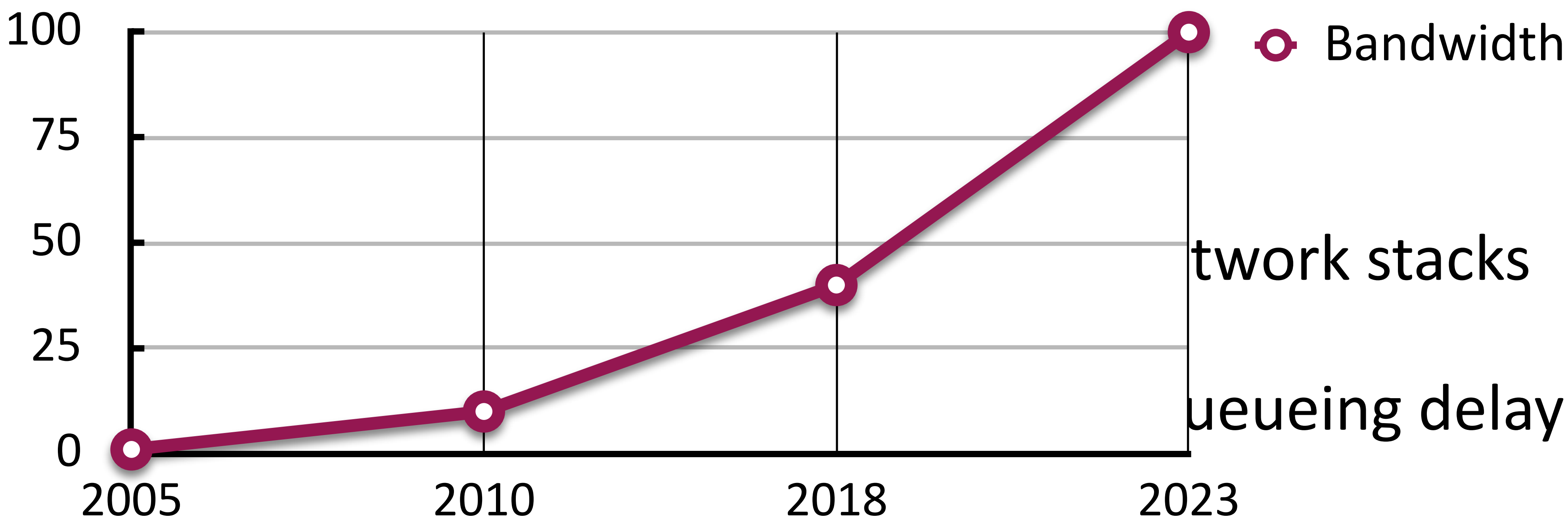
Where is the puck going?



Tape is dead,
Disk is tape,
SSD is disk,
RAM is the king!

- Jim Gray

Network Technology Trends



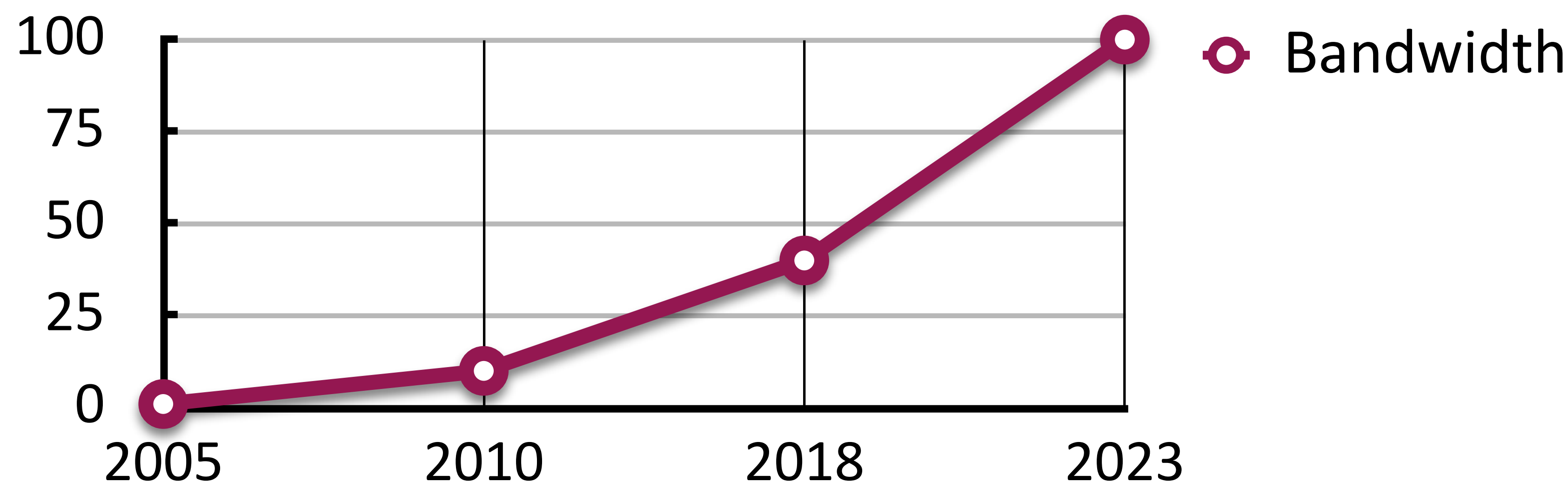
twork stacks

ueueing delay

Powerful implications

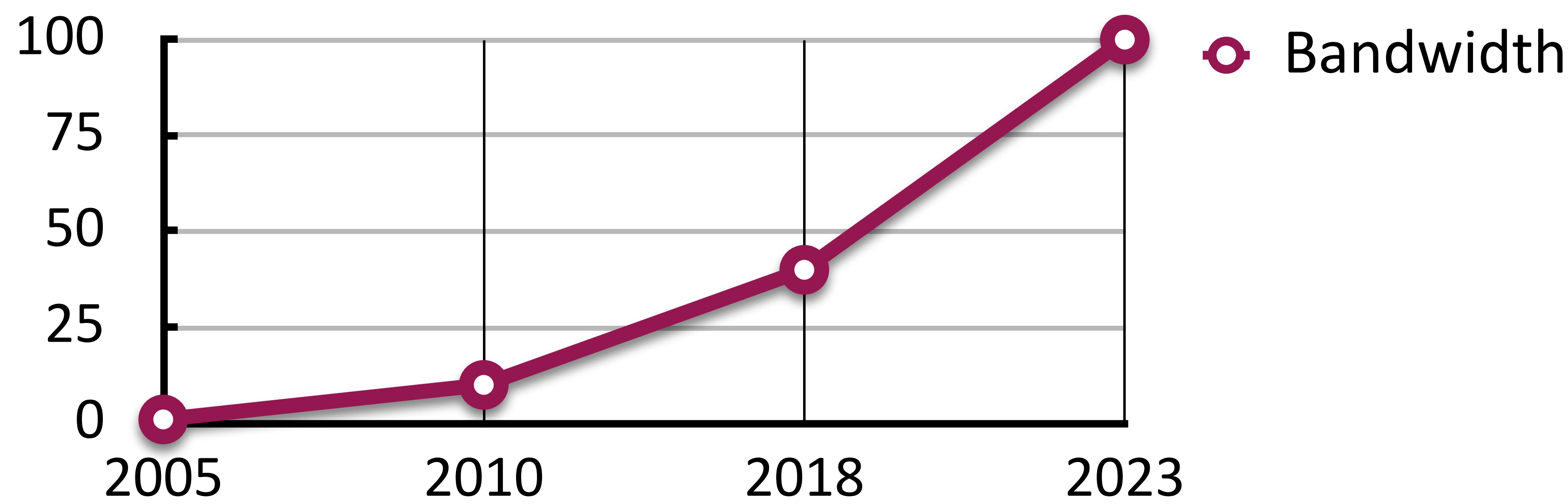
- Remote memory faster than local SSD
  - When queueing delay = 0

Unsustainable CPU overheads



- **Existing network stacks were designed for 1Gbps networks**
  - Known TCP problem: ~3.2Gbps per core
  - With low-level optimizations: ~9-12Gbps per core
    - 40Gbps would take >3 cores per server!
    - **100Gbps would take >8 cores per server!!**
- **Take away: unsustainable cloud economics**
  - Every core used for the stack is a core stolen from applications/ customers

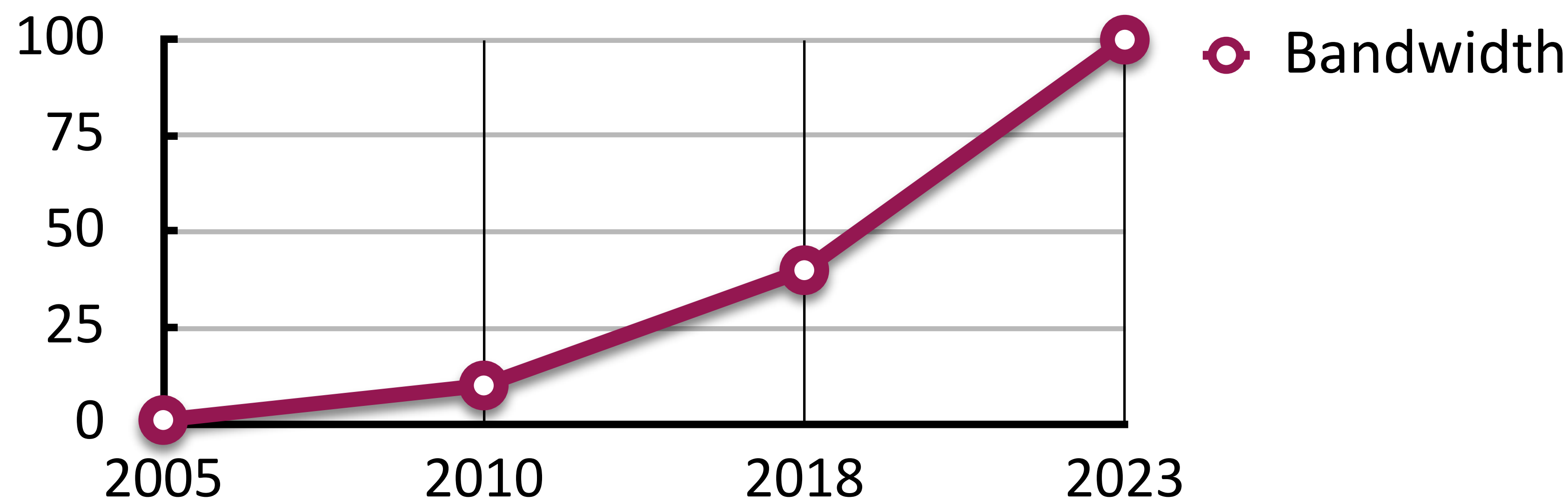
Curse of queueing delay



	~2005 (1Gbps)		2018 (40Gbps)	
	Latency (us)	%	Latency (us)	%
TOTAL	**18.92**		**6.30**	
Queueing (4MB buffers, 64 ports)	**488.3 (per congestion point)**		**12.21 (per congestion point)**	
Propagation delay	0.88	5	0.88	13
Transmission delay	11.44	61	0.29	5
TOTAL	**18.92**		**6.30**	
Queueing (4MB buffers, 64 ports)	**488.3 (per congestion point)**		**12.21 (per congestion point)**	

- **Take away: queueing delay is the core bottleneck**
  - **End-to-end latency bottlenecked by queueing delay**

Remote Memory Faster than Local Storage



- **Under zero queueing:**
  - Remote memory access takes less than 6.3us
  - Local SSD access latency today is 25us (hardware, ignoring stack)
  - Remote Direct Memory Access (RDMA) becomes feasible
  - **However, RDMA requires lossless network fabric**
    - Known problem with RDMA over Ethernet: congestion collapse

- **Take away: RDMA applicability limited by drops in network fabric**

Current Network Stacks are the Bottleneck!

- **Lot of research in “hardware offload”**
  - Implementing TCP (and other mechanisms) on hardware
  - Lots of interesting challenges
- **Lot of research in low-latency transport design**
  - TCP was not designed for low latency
  - New transport protocols for ultra low-latency
- **Lot of research in kernel-bypass**
  - TCP requires processing each and every packet
  - 1Gbps links: 90,000 packets per second
  - 100Gbps links: 9 million packets per second
  - Extremely high CPU requirements
  - Bypass the kernel entirely
    - Implement congestion control in user space, in hardware?

Closing Thoughts

- **These are exciting times for computer networking**
    - The first ever since the invention of the Internet
    - You are witness to the transformation!!!!
- **And, I am glad I got the chance to introduce you to this world :-)**
    - You have made me a better teacher!!!!
    - Thank you.
- **Wherever you end up:**
    - Please remember me
    - Say hello if you see me
    - Remember, there is nothing more important than
        - **Knowing the fundamentals!!!!**
        - **Being happy!!!!**