What Can a Software Company Bring to Chiplets?

Paul Borrill, Founder/CEO

& Team:

Steve, Liane, Charlie, Melissa, Matt, Susan, James, Nik

January 25, 2023. - V1.3a

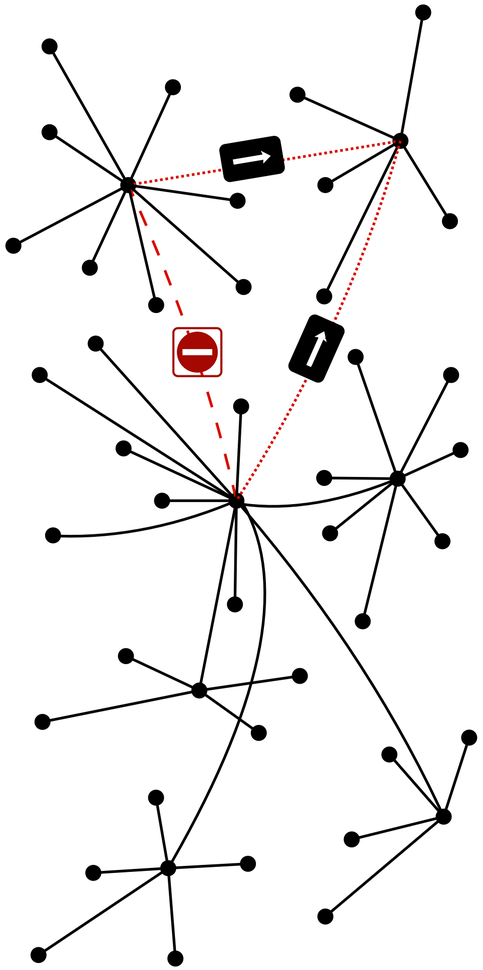
DAEDAELUS

CHIPLET SUMMIT

Daedaelus

- A graph ***software*** company focused on dependable computing
- We solve what some folks consider unsolvable problems in the communication between pieces of a distributed application
  - Which reside on different computers
  - Which communicate over a fallible network
  - Which require agreement on certain facts in order to operate correctly
- Incidental to our solution, we write code for an FPGA NIC
- Incidental to our solution, we use a mesh network of chiplet servers

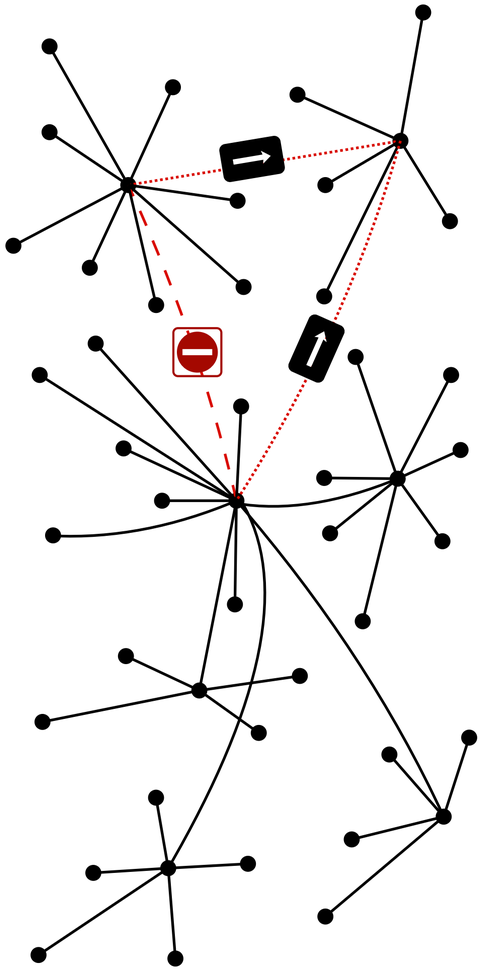
Application errors caused by communication issues



- 80% of failures have a **catastrophic impact**, with data loss being the most common (27%)
- 90% of the **failures are silent**, the rest produce warnings that are unclear
- 21% of the failures lead to permanent damage to the system.
- This **damage persists** even after the network partition heals

https://dl.acm.org/doi/10.1145/3576192

Application errors caused by communication issues



Partial Network Partitioning

BASIL ALKHATIB, University of Waterloo, Canada
SREEHARSHA UDAYASHANKAR, University of Waterloo, Canada
SARA QUNAIBI, University of Waterloo, Canada
AHMED ALQURAAN, University of Waterloo, Canada
MOHAMMED ALFATAFTA, University of Waterloo, Canada
WAEL AL-MANASRAH, University of Waterloo, Canada
ALEX DEPOUTOVITCH, Huawei Research Canada, Canada
SAMER AL-KISWANY, University of Waterloo, Canada

We present an extensive study focused on partial network partitioning. Partial network partitions disrupt the communication between some but not all nodes in a cluster. First, we conduct a comprehensive study of system failures caused by this fault in 13 popular systems. Our study reveals that the studied failures are catastrophic (e.g., lead to data loss), easily manifest, and are mainly due to design flaws. Our analysis identifies vulnerabilities in core systems mechanisms including scheduling, membership management, and ZooKeeper-based configuration management.

Second, we dissect the design of nine popular systems and identify four principled approaches for tolerating partial partitions. Unfortunately, our analysis shows that implemented fault tolerance techniques are inadequate for modern systems; they either patch a particular mechanism or lead to a complete cluster shutdown, even when alternative network paths exist.

Finally, our findings motivate us to build Nifty, a transparent communication layer that masks partial network partitions. Nifty builds an overlay between nodes to detour packets around partial partitions. Nifty provides an approach for applications to optimize their operation during a partial partition. We demonstrate the benefit of this approach through integrating Nifty with VoltDB, HDFS, and Kafka.

CCS Concepts: • **Computer systems organization** → **Cloud computing**; **Reliability**; **Availability**; • **Networks** → **Network reliability**.

Additional Key Words and Phrases: network failures, fault tolerance, partial network partitions, distributed systems, reliability.

1 INTRODUCTION

Modern networks are complex. They use heterogeneous hardware and software [1], deploy diverse middleboxes (e.g., NAT, load balancers, and firewalls) [2–4], and span multiple data centers [2, 4]. Despite the high redundancy built into modern networks, catastrophic failures are common [1, 3, 5, 6]. Nevertheless, modern cloud systems

Just Accepted

Authors' addresses: Basil Alkhatib, b2alkhatib@uwaterloo.ca, University of Waterloo, Canada; Sreeharsha Udayashankar, sreeharsha.udayashankar@uwaterloo.ca, University of Waterloo, Canada; Sara Qunaibi, squnaibi@uwaterloo.ca, University of Waterloo, Canada; Ahmed Alquraan, ahmed.alquraan@uwaterloo.ca, University of Waterloo, Canada; Mohammed Alfatafta, m.alfatafta@uwaterloo.ca, University of Waterloo, Canada; Wael Al-Manasrah, wael.al-manasrah@uwaterloo.ca, University of Waterloo, Waterloo, Canada; Alex Depoutovitch, alex.depoutovitch@huawei.com, Huawei Research Canada, Canada; Samer Al-Kiswany, University of Waterloo, Canada, salkiswany@uwaterloo.ca.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2022 Association for Computing Machinery.

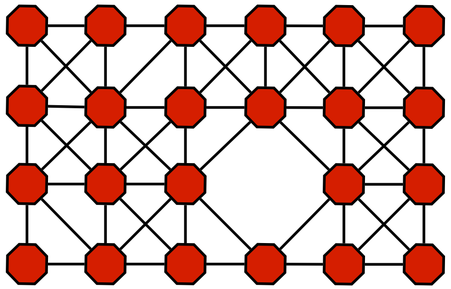
0734-2071/2022/12-ART $15.00
https://doi.org/10.1145/3576192

https://dl.acm.org/doi/10.1145/3576192

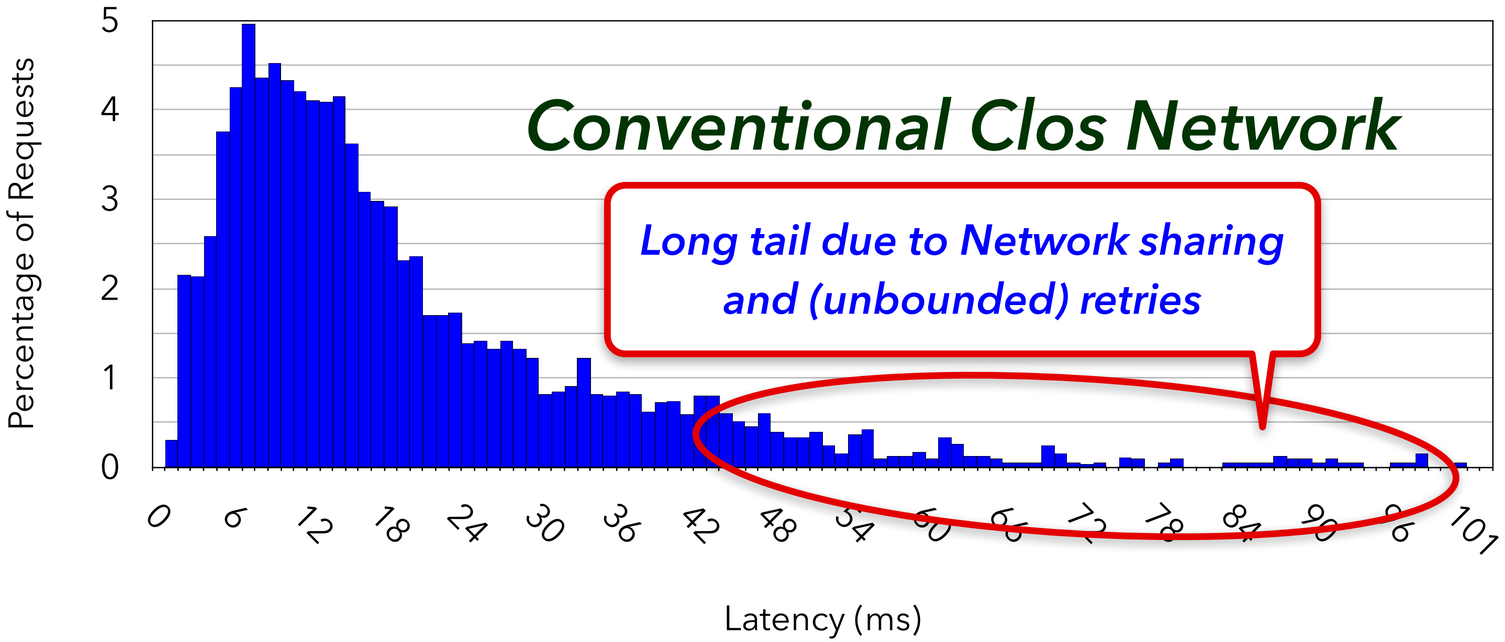
Tail Latency

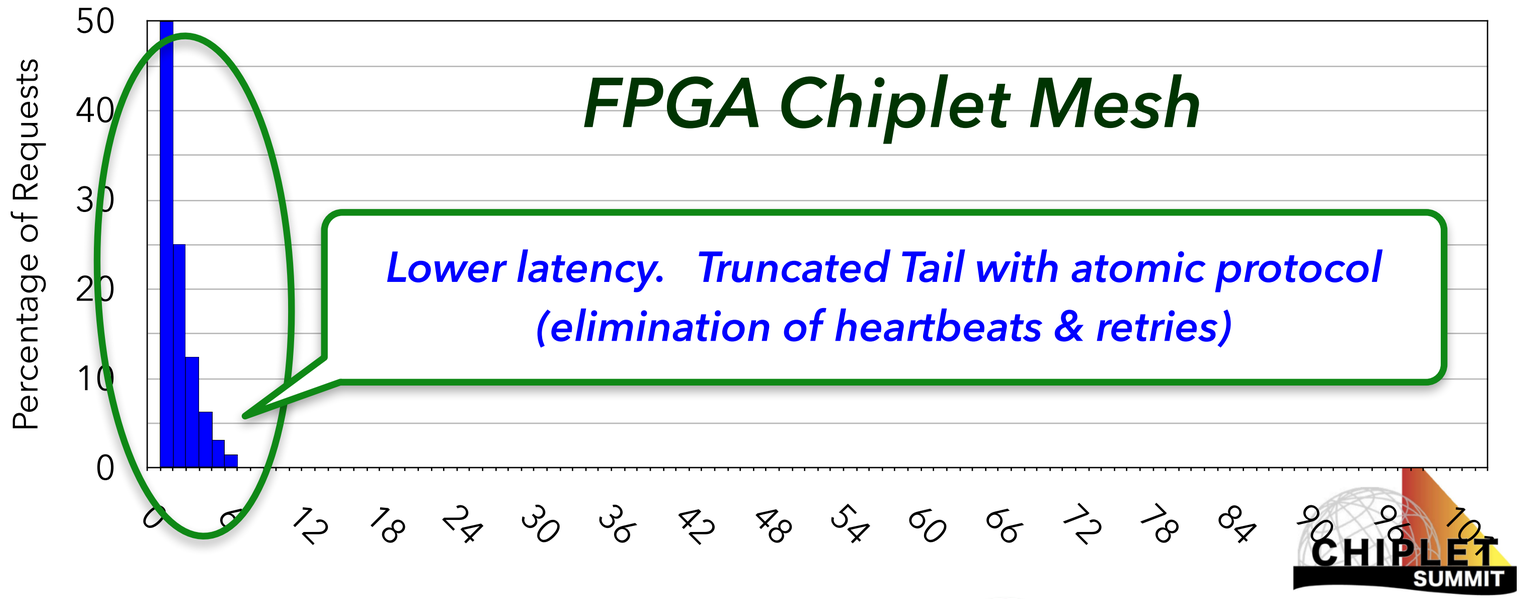
Daedaelus reduces latency in ways conventional networks cannot:

- Direct connections
- Multicast consensus, in parallel over 8 ports instead of serial over 1
- Truncated Tail Latency – protocol knows it failed or succeeded (without heartbeats or timeouts)



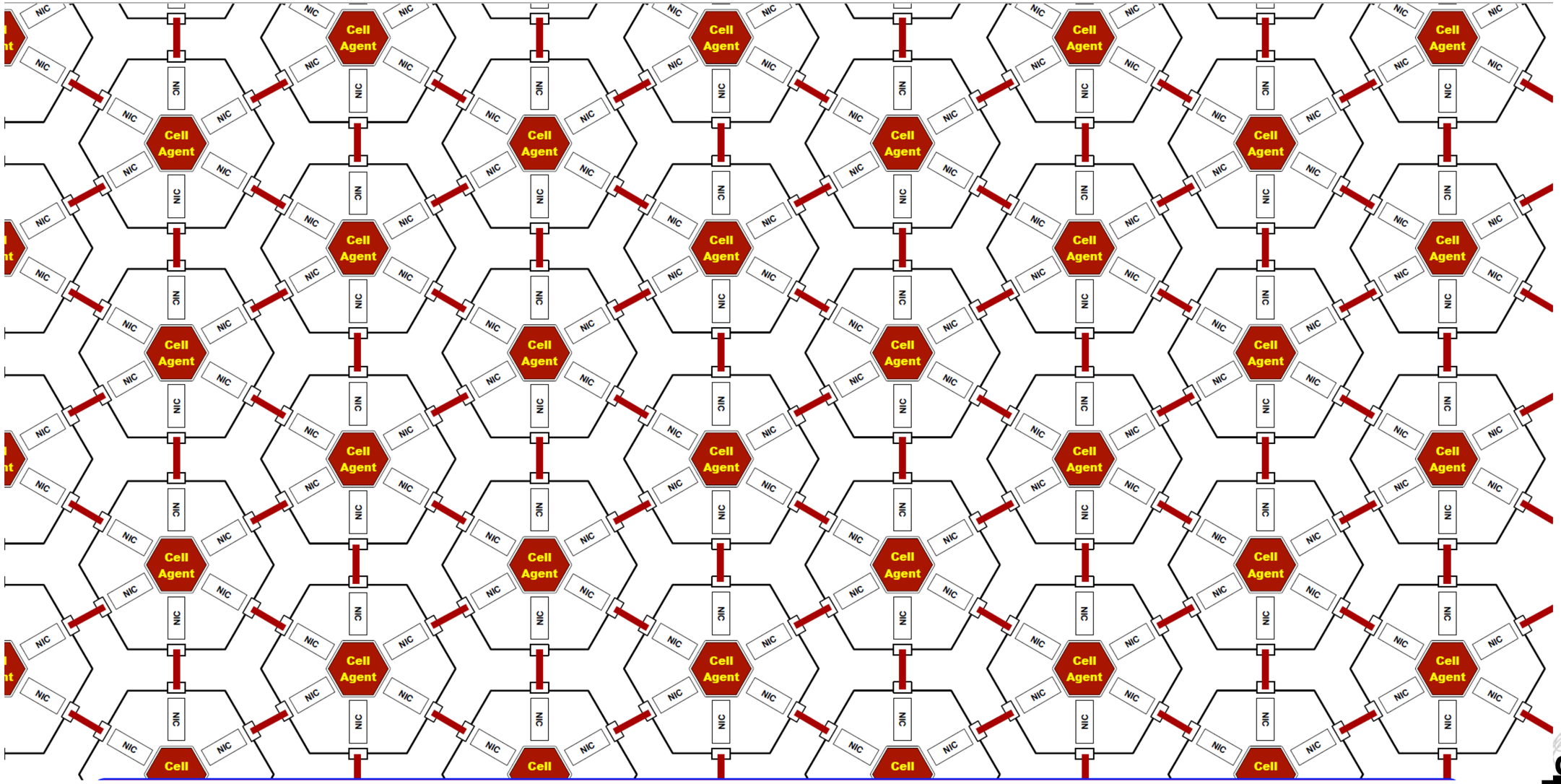
Fallible Chiplet Mesh





Plane (mesh) of servers

not the network tree we all assume



Daedaelus changes basic assumptions

- There are no network switches within our mesh
  - Don’t worry, the uplinks from the mesh work just like you’re used to
- There are no dropped packets
  - If a link fails, traffic is paused while we re route, locally, around the failure
  - If a packet doesn’t reach its destination, we know right away
  - If there is ambiguity about whether a packet arrived or not, during the route around we communicate with both ends and ensure both ends have the same facts about whether that packet was delivered or not
  - There are ugly corner cases. We design them out. That’s what we do.
- At the distributed application level, this allows actual agreement on facts across a set of nodes, even though the CAP theorem “proves” that can’t be done reliably,
  we circumvent the CAP proof’s assumptions.

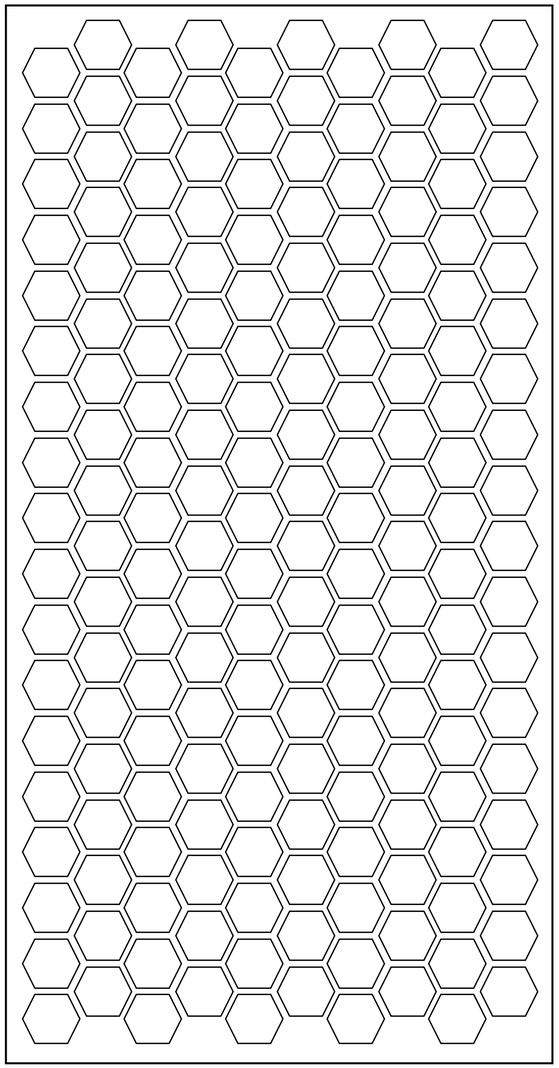
Daedaelus Evolves to need Chiplets

- Daedaelus' initial product will use as a node a server, with a PCIe slot, which has an FPGA card in it, which has 6 to 8 cables connected to its "neighbors" in the "plane".
  - We will do early work with teams which write distributed software, tune our algorithms, and develop our APIs for tighter application integration on this platform
  - A rack or row of these will be a bit more expensive than best practice standard data center deployment today. But it's different, and eventually infrastructure teams will push back on deploying more of our configuration.
- That's where people in this audience come in

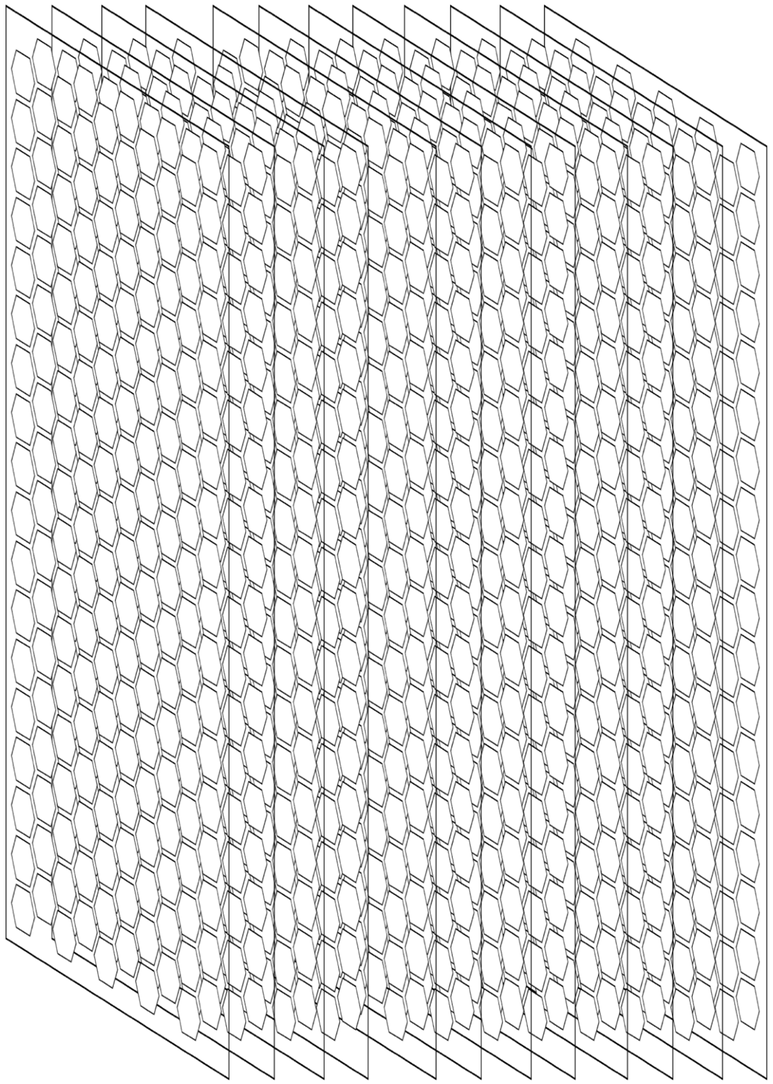
While we were solving a software problem we accidentally invented the module network

- I want you to picture a complete small server on a single module
  - Perhaps 10cm on a side, square (extra credit for hexagon)
  - CPU chiplet, FPGA chiplet for our NIC, flash controller and flash, DRAM
  - I guarantee you there are people in this room ahead of us on this
- Take a 3 x 6 foot sheet of steel
  - Tile a 10x20 array of 200 modules, side by side, on the sheet of steel
  - Sink the heat into the steel (someone here can do this better than me)
  - Put power busbars under the steel (again someone here can do better)
  - Connect 2cm cables between modules, side to side, up and down, and diagonally. The hexagon offers an alternative here with 6 connections.
  - Mount 10 of these sheets of steel inside a rack.
    - Better, skip the rack and mount 12 in the space the rack used to need

Rack Scale



200 module scale servers
on a 3 x 6-foot sheet of steel



12 such sheets of steel
in the space of a rack

Stop and think about what this means

- We just designed the motherboard layer out of servers entirely
- We also turned the entire I/O system into chiplets – software defined interconnect
- This is a structural cost reduction relative to “best practice” servers of the last 22 years
- Large memory is going to require pooling memory across module servers, with implications for latency and probably requiring evolution of CXL and of large memory applications
- True peer to peer communication over CXL without switches will probably require evolution of CXL as well
- There are people in this room who understand all this better than I do

Next Steps

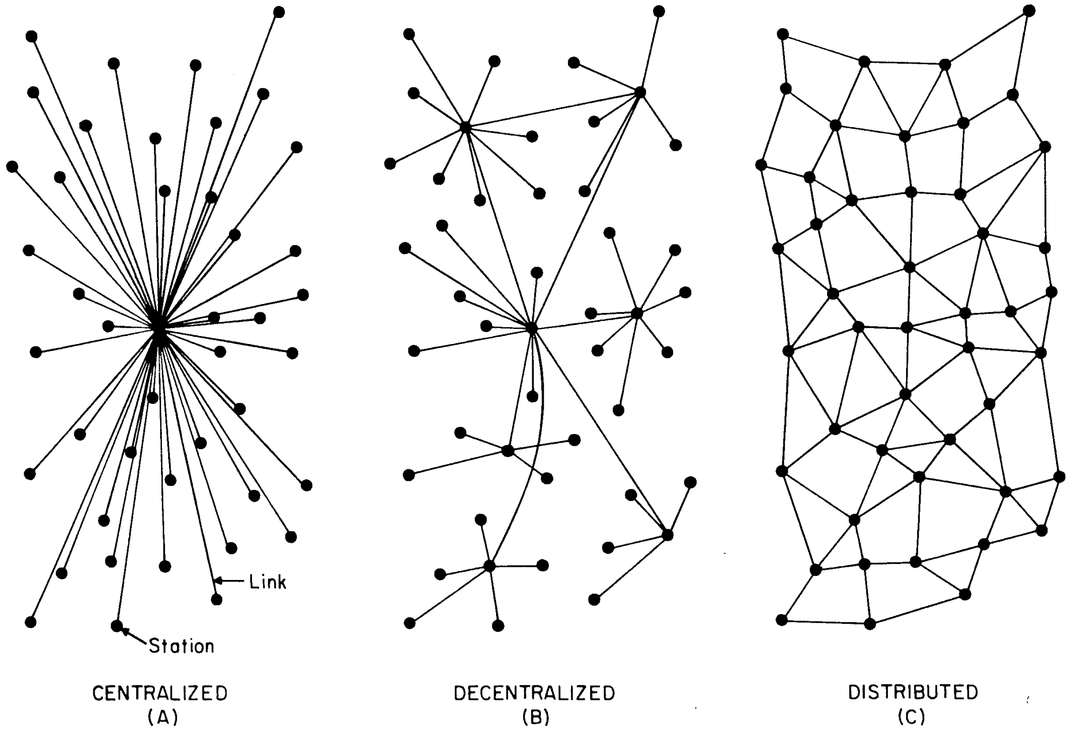
- Daedaelus is pre funding
  - We've spent a very long time thinking through the distributed software communication robustness problem and…umm…have the scars to prove it can't be fixed without radical network topologies and control of the NIC.
  - Time to raise, write code, partner with customers, make mistakes and learn
- We need to intersect with a mesh hardware platform later this year
  - That gives us time to develop and tune our FPGA code
  - And lets us focus on the needs of distributed software developers
  - We'd need a much bigger raise, more talent, and a longer runway to build modules and a row scale platform ourselves
- Does the problem resonate with you? It takes a community and we would love to connect.
- Engineers, Customer Reps, and Investors… Please follow us on twitter (@bozdog), and join our newly created discord server by email us at info@Daedaelus.com for an invite to our newsletter

Application and Deployment Enablers

- These collections of servers can be managed in graphs and sets, not individually. We think this enables much larger server counts.
  - Network ACLs are derived from those graphs and sets, not manual
  - A device cannot form the address of a node outside its graph(s)
- A network node could fail, be re instantiated somewhere else in the same mesh, and resume network communication with no dropped or duplicated packets. We're still thinking through what the application would have to do to actually resume correctly.

Combining Chiplets in Software?

Demo



Mathematica Demo of Resilient Baran Graphs on Chiplet Modules

The Algebraic connectivity of a graph is the numerically second smallest eigenvalue (counting multiple eigenvalues separately) of the Laplacian matrix of a graph. In other words, it is the second smallest root of the graph's Laplacian polynomial. This eigenvalue is greater than 0 iff is a connected graph.

Results

Chiplets per Module	Modules per Sheet	Total Servers	Func. Links	Spanning Trees
2	20	40	96	4.2597×10^{20}
4	20	80	250	1.2308×10^{54}
6	20	120	404	8.8277×10^{86}
8	20	160	558	5.0348×10^{119}
10	20	200	712	2.674×10^{152}
12	20	240	866	1.3804×10^{185}
14	20	280	1020	7.035×10^{217}
16	20	320	1174	3.5631×10^{250}
18	20	360	1328	1.799×10^{283}
20	20	400	1482	9.0689×10^{315}
22	20	440	1636	4.5679×10^{348}
24	20	480	1790	2.2998×10^{381}
26	20	520	1944	1.1577×10^{414}
28	20	560	2098	5.8266×10^{446}
30	20	600	2252	2.9324×10^{479}
32	20	640	2406	1.4758×10^{512}
34	20	680	2560	7.4268×10^{544}
36	20	720	2714	3.7375×10^{577}
38	20	760	2868	1.8809×10^{610}
40	20	800	3022	9.4655×10^{642}
42	20	840	3176	4.7635×10^{675}

Thank You

Paul Borrill
Founder/CEO and Team
paul@daedaelus.com

DÆDÆLUS

CHIPLET SUMMIT