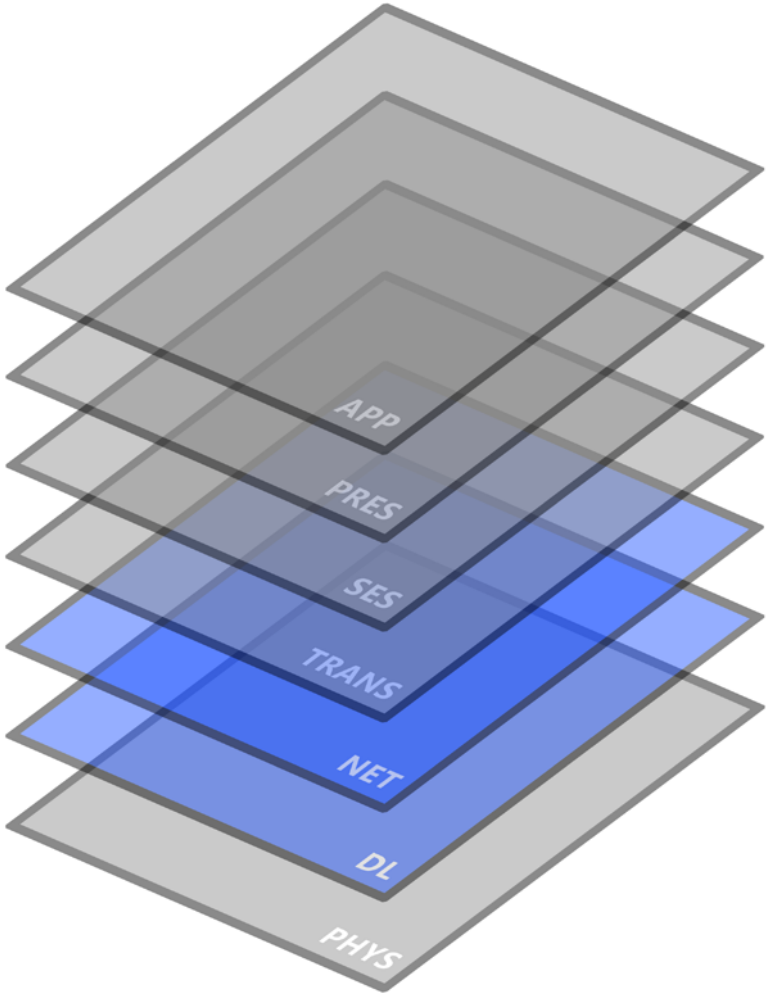
IronStack: security applications for software-defined networks

Zhiyuan Teo

It’s a work in progress. Don’t assume I got everything right.

OSI model of networking



L7: Application

L6: Presentation

L5: Session

L4: Transport

L3: Network

L2: Data link

L1: Physical

What is software-defined networking?

- Traditional switches run their own hardware algorithms for routing, with very little flexibility for customization.

- Idea: separate data plane from the control plane.

- Why only now?

- Another example of separating policy from mechanism.

- Policy: routing algorithms.

- Mechanism: hardware transmission.

- Typically located at L2/L3.

OpenFlow

- Background.

- It's a protocol for a controller to access the forwarding plane of a switch.

- How?

- Quite critically: can also forward packets from the data plane to the control plane.

- Flows can be created for specific actions.

What's a 'flow'?

- Describes aggregate movement of data that match certain packet fields.

- A 12-tuple uniquely identifies a flow entry:

  ingress port
  ethernet src, ethernet dst, ethertype
  VLAN id, VLAN priority
  IP src, IP dst, IP proto, IP ToS
  TCP/UDP src port, TCP/UDP dst port

- Packets matching a given 12-tuple will be considered part of that flow.

- Wildcards are allowed in flow descriptions.

- eg. (*, *, *, *, *, *, 10.0.0.1, 10.0.0.2, TCP, *, *, 22)

Flow actions

- Each flow can be associated with one or more actions.

- Possible actions:

  flood packet to all switch ports.
  forward packet out specific switch port/controller.
  drop packet.
  modify fields in packet.

- Suppose (*, *, *, *, *, *, 10.0.0.1, 10.0.0.2, TCP, *, *, 22) is associated with the action ‘drop packet’.

- SSH communications will not be possible from 10.0.0.1 to 10.0.0.2.

- Implemented on the switch, so clients don’t know about them.

What else? A taster of OpenFlow capabilities

- Inspect packets traversing the switch.

- Output arbitrary packets onto switch.

- Redirect flow to different switch port.

- Rewrite src/dest IP and port, MAC address, etc.

- Manipulate link state associated with a port.*

- Track changes in switch state.

- Keep statistics about switch and flow usage.

Some applications of OpenFlow

- Traffic engineering @ Google
http://www.opennetsummit.org/archives/apr12/hoelzle-tue-openflow.pdf

- Network Invariant tracking (VeriFlow)

- Composing Software Defined Networks

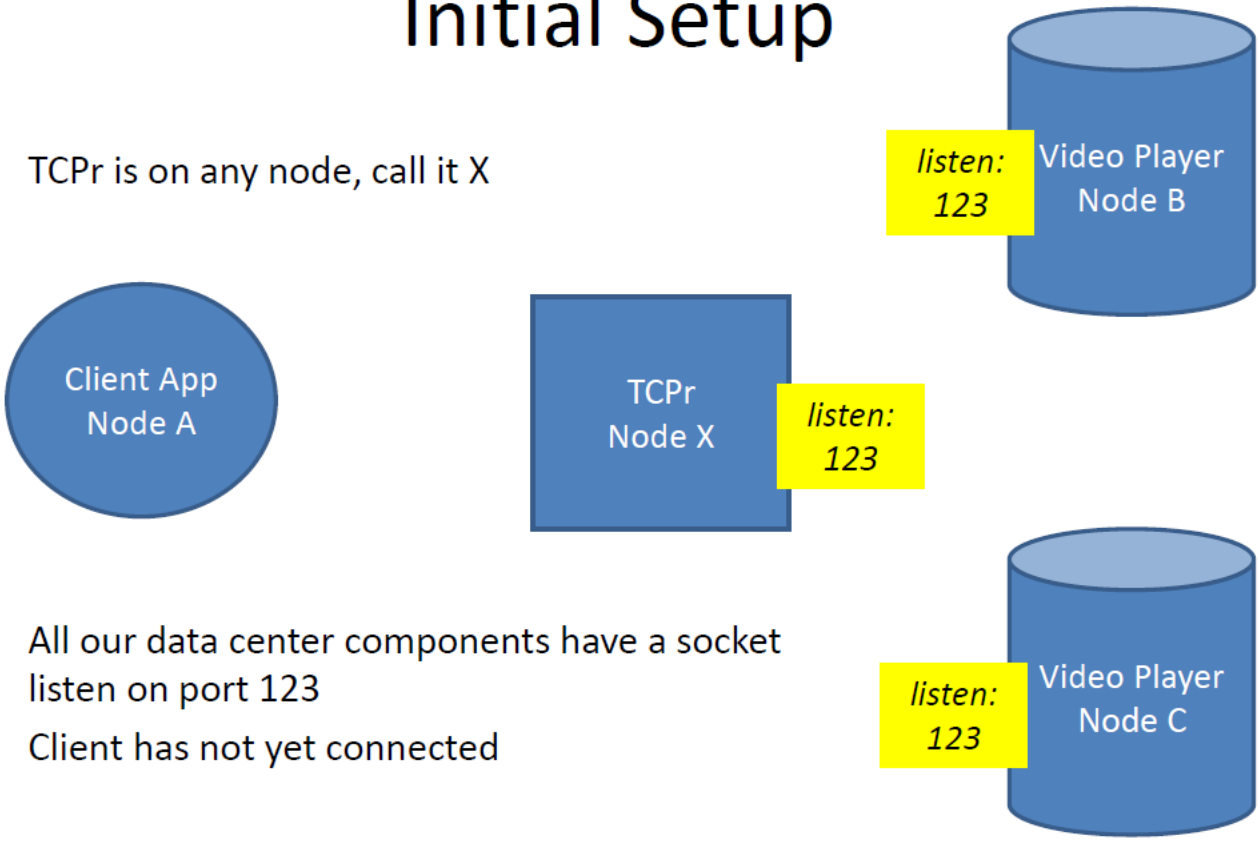
- TCP-R

It is a hot topic at NSDI!

TCP-R (developed at Cornell)

Initial Setup

- TCPr is on any node, call it X



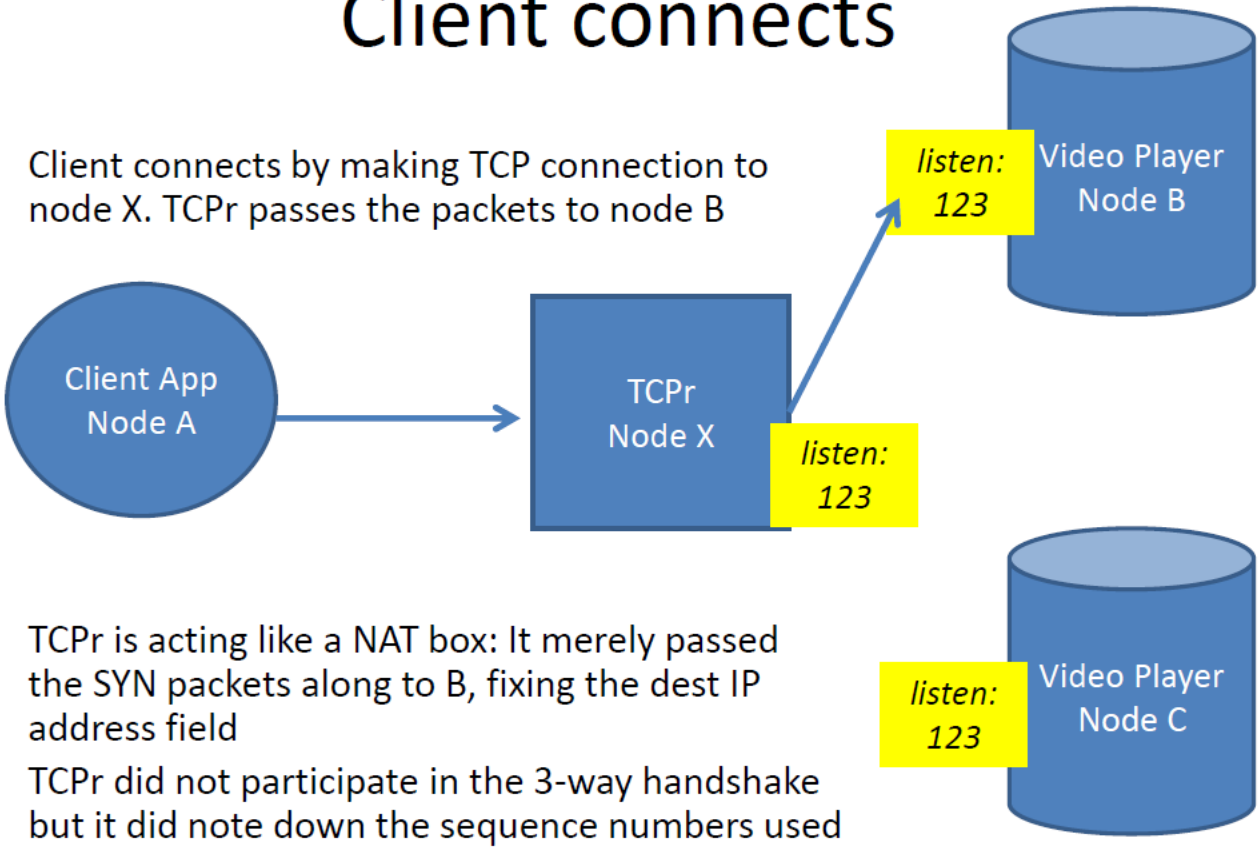
- All our data center components have a socket listen on port 123
- Client has not yet connected

Slides courtesy of Ken Birman

TCP-R (developed at Cornell)

Client connects

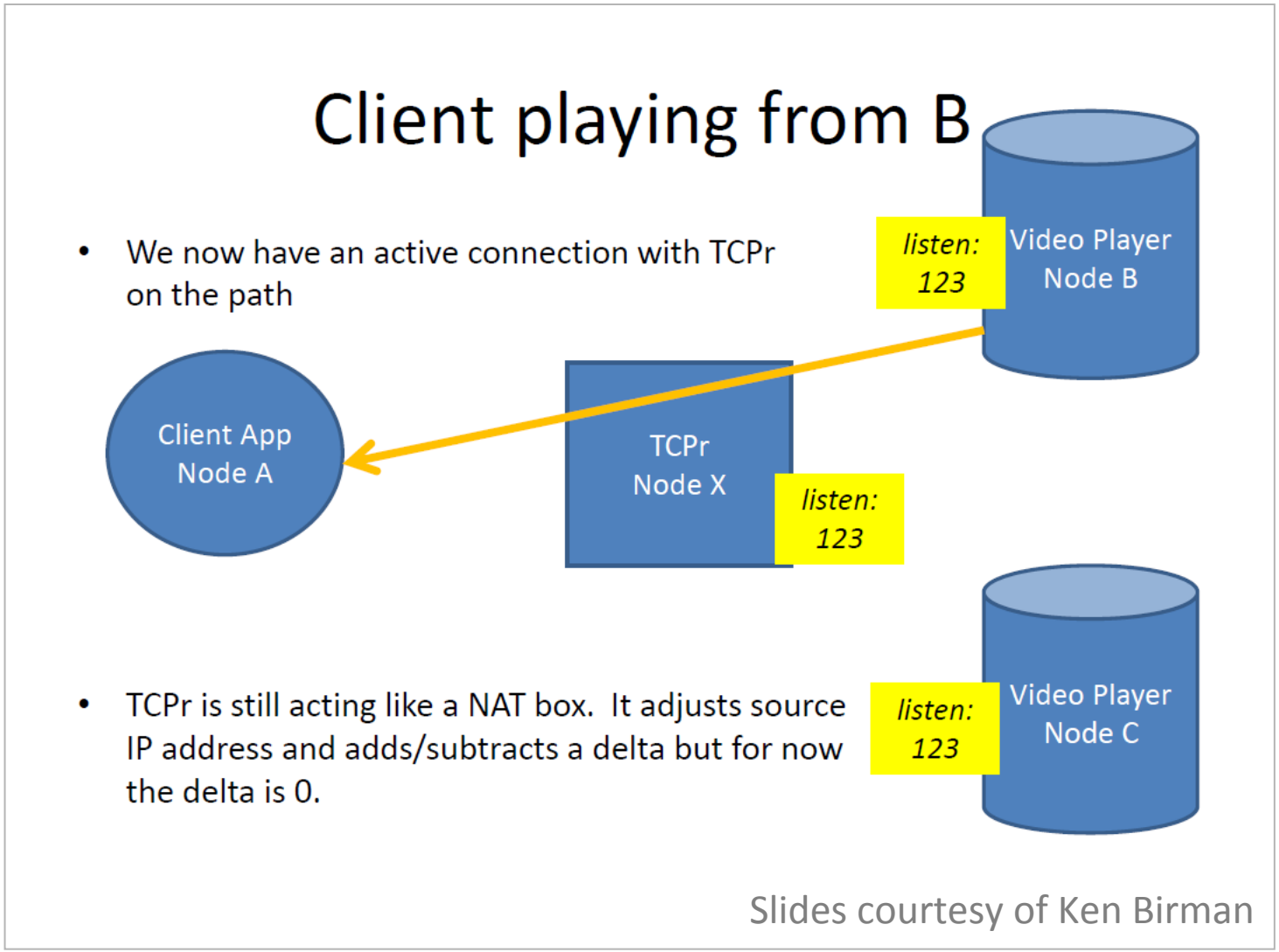
- Client connects by making TCP connection to node X. TCPr passes the packets to node B



- TCPr is acting like a NAT box: It merely passed the SYN packets along to B, fixing the dest IP address field
- TCPr did not participate in the 3-way handshake but it did note down the sequence numbers used

Slides courtesy of Ken Birman

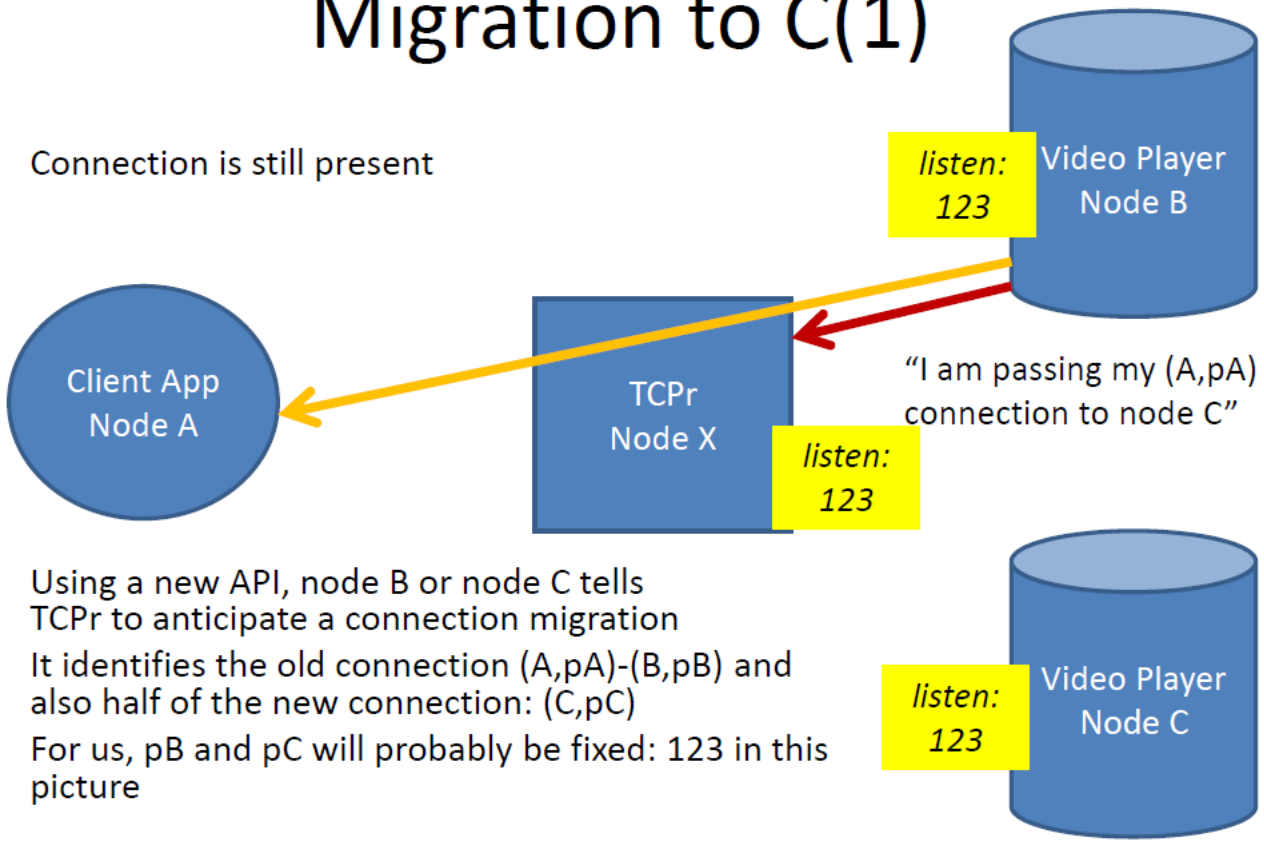
TCP-R (developed at Cornell)



TCP-R (developed at Cornell)

Migration to C(1)

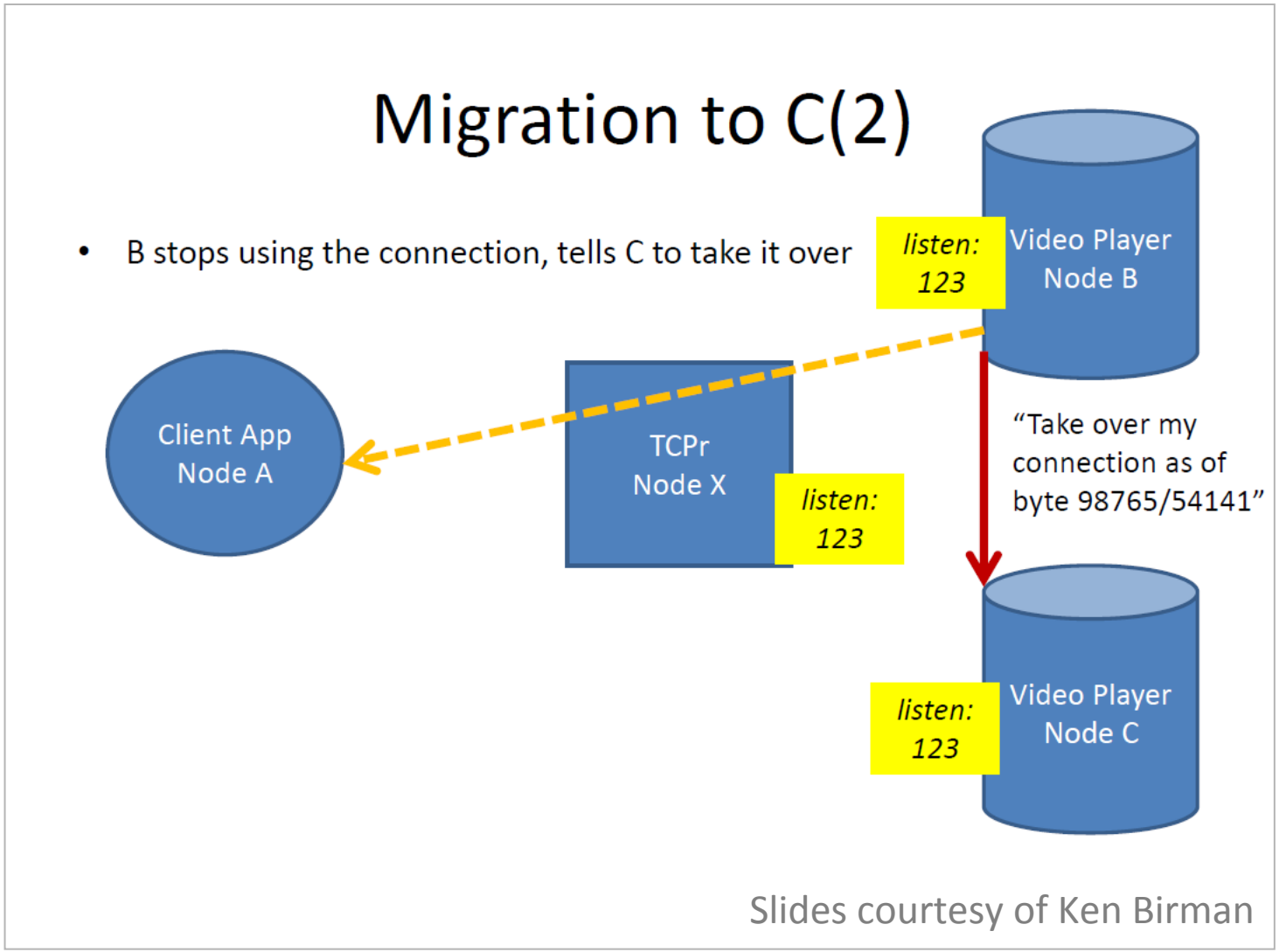
- Connection is still present



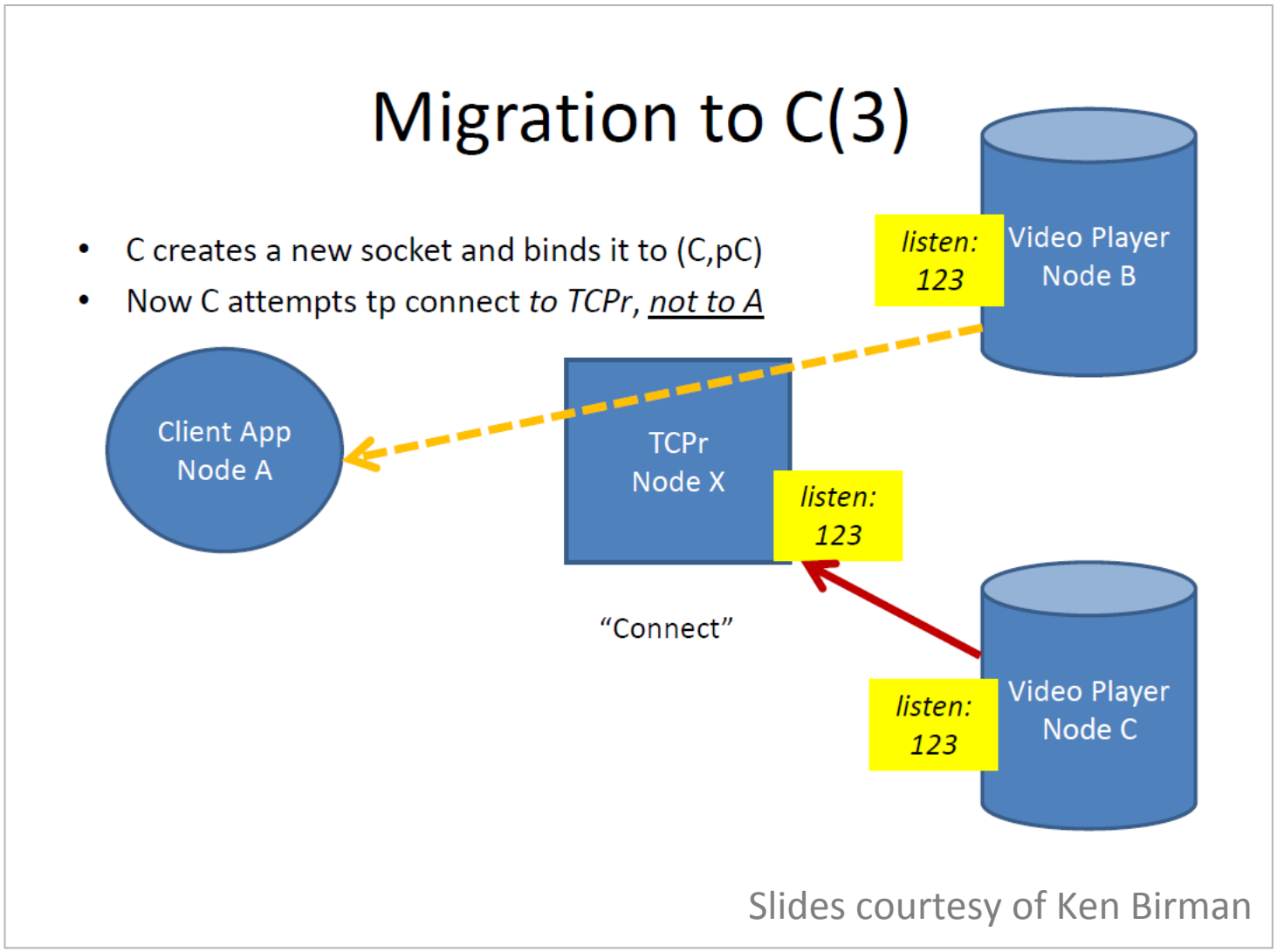
- Using a new API, node B or node C tells TCPr to anticipate a connection migration
- It identifies the old connection (A,pA)-(B,pB) and also half of the new connection: (C,pC)
- For us, pB and pC will probably be fixed: 123 in this picture

Slides courtesy of Ken Birman

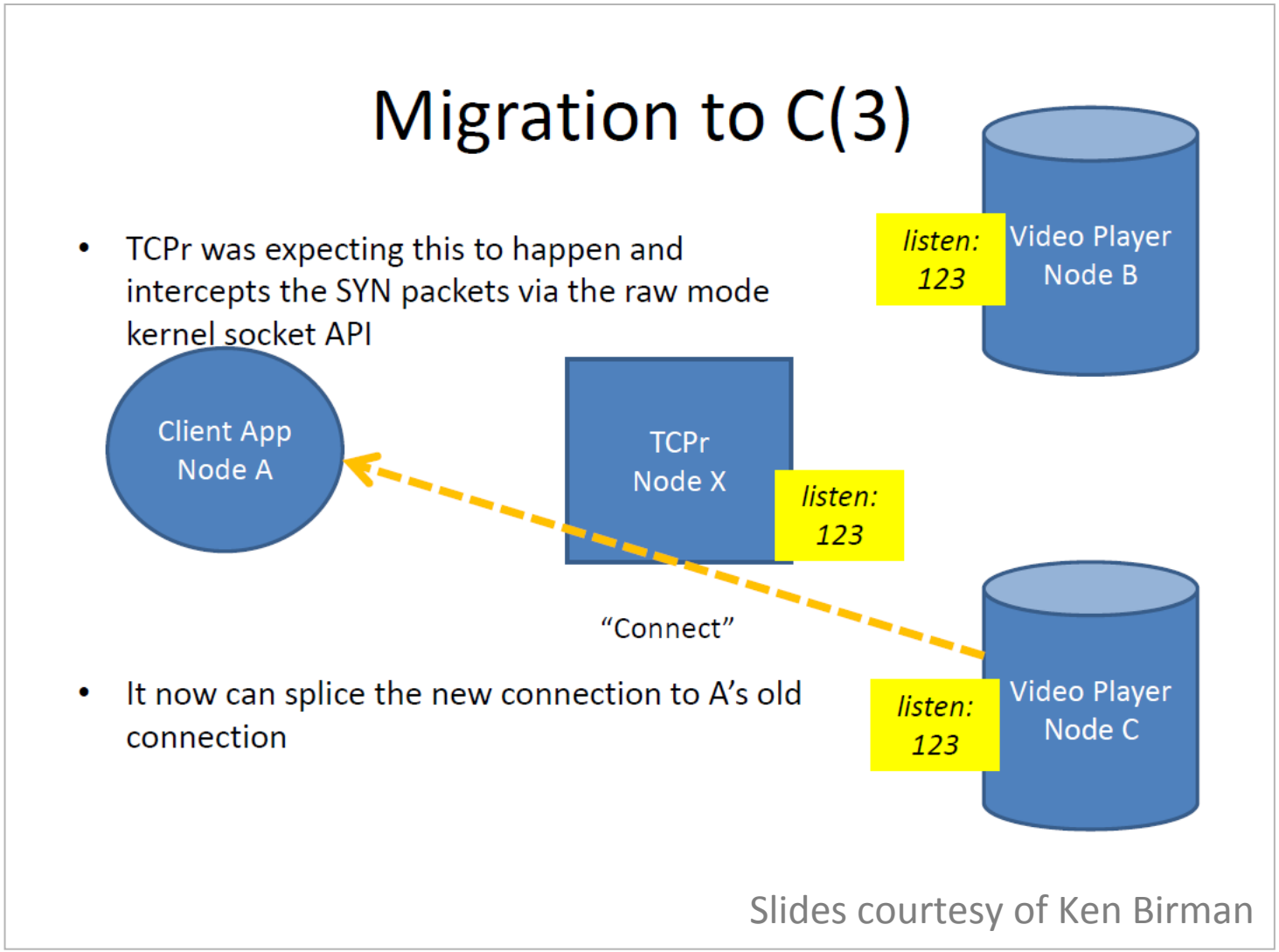
TCP-R (developed at Cornell)



TCP-R (developed at Cornell)



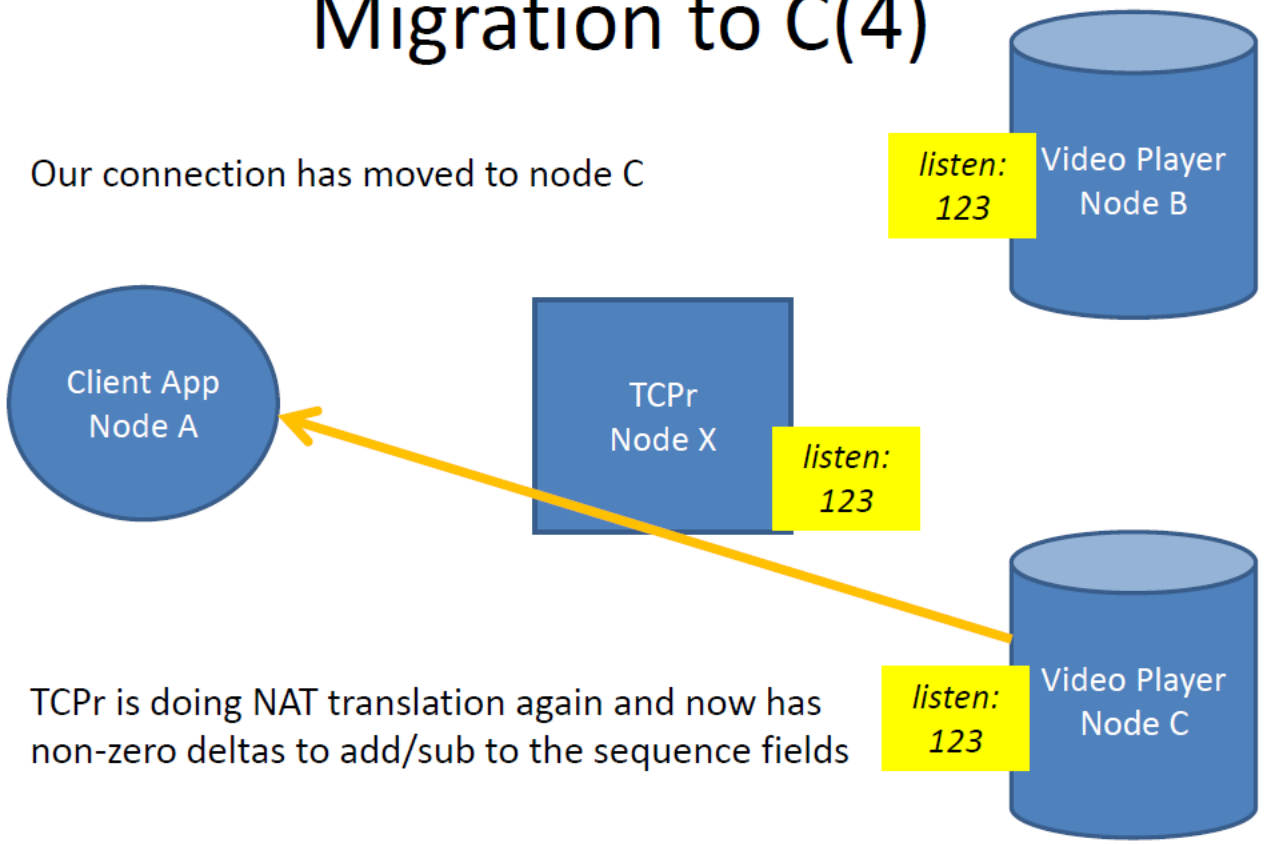
TCP-R (developed at Cornell)



TCP-R (developed at Cornell)

Migration to C(4)

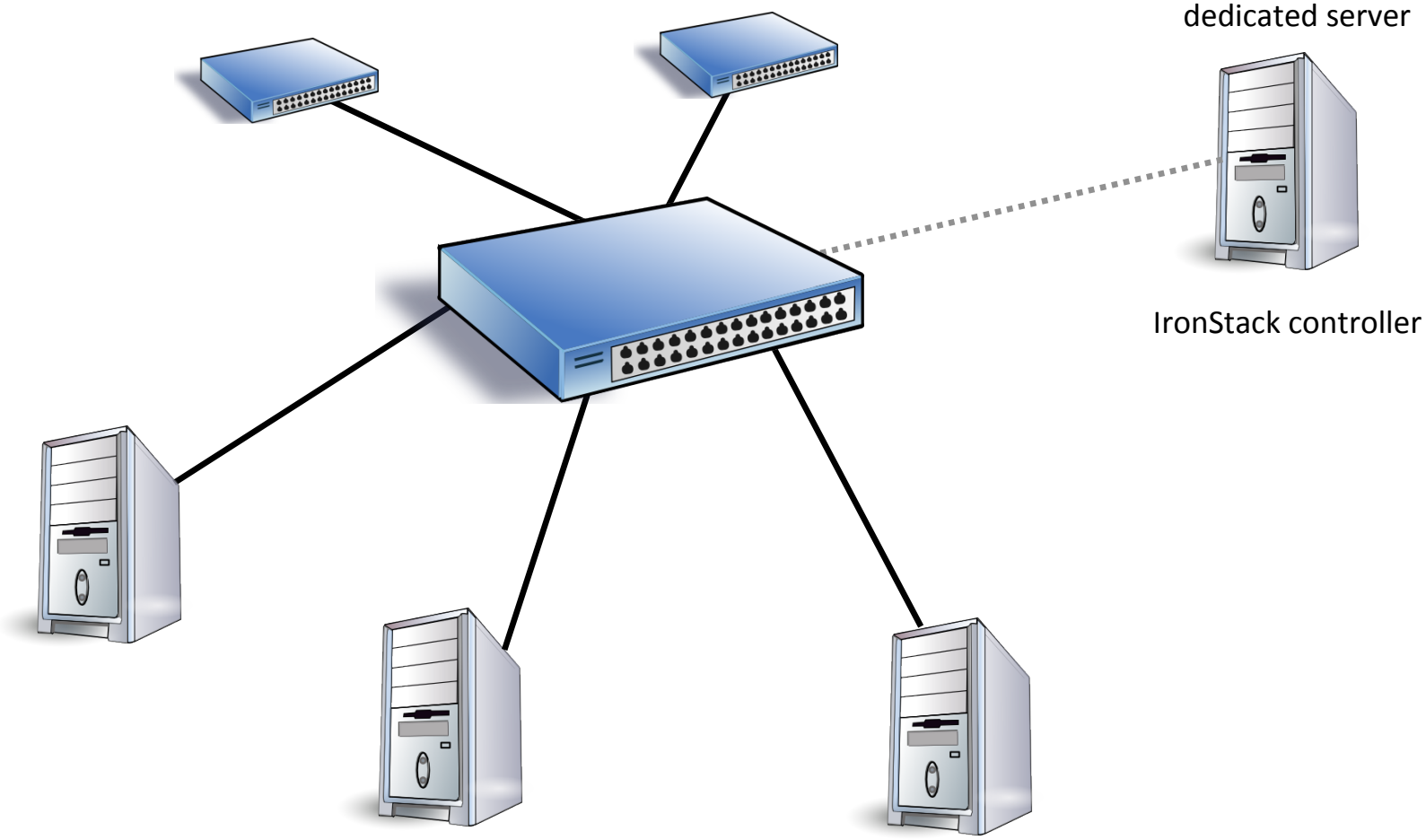
- Our connection has moved to node C



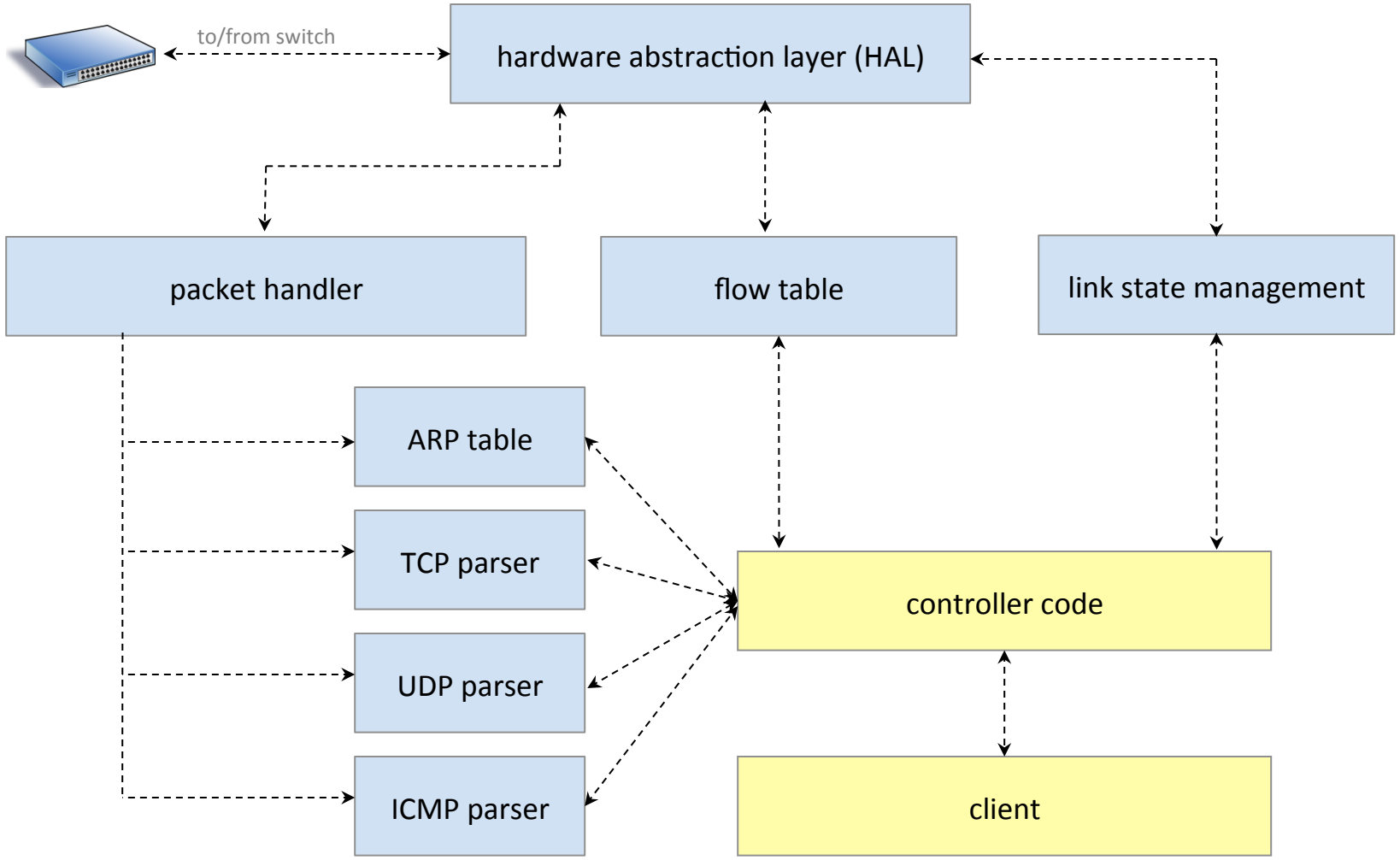
- TCPr is doing NAT translation again and now has non-zero deltas to add/sub to the sequence fields

Slides courtesy of Ken Birman

IronStack hardware architecture



IronStack software architecture



How does a client talk to the local IronStack controller?

- Previous diagram is a little misleading: clients do not sit on the same machine as IronStack.

- Actually 2 problems: how does a client know that there is a local controller AND how does it talk to the controller?

- Safety: packets sent out onto the switch must reach controller, but otherwise be dropped if controller doesn't exist/is dead.

How does a client talk to the local IronStack controller?

- IronStack controller must be able to distinguish special packets from regular traffic.

- Can't use magic numbers for any of the fields, they could be in use.

- Solution: use a hack. Send packets with src eth 0, dest eth 0, src IP 0.0.0.0 and dest IP 0.0.0.0. Use flows on switch to capture this traffic and forward to controller.

- Packet is not routable, so any normal switch will drop it.

- Problem: not all kernels allow packets to be sent with src/dst IP 0.0.0.0. Some also don't let you change the MAC address.

Let’s see some applications.

IronGuard

- Forward TCP packets to a special module on a special system.

- Can reconstruct entire TCP transaction between endpoints.

- Do a ‘man-in-the-middle’ defense!

- How?

IronGuard

- All flows on TCP src/dest port 21 will be forwarded to a special switch port other than the one connected to the endpoint.

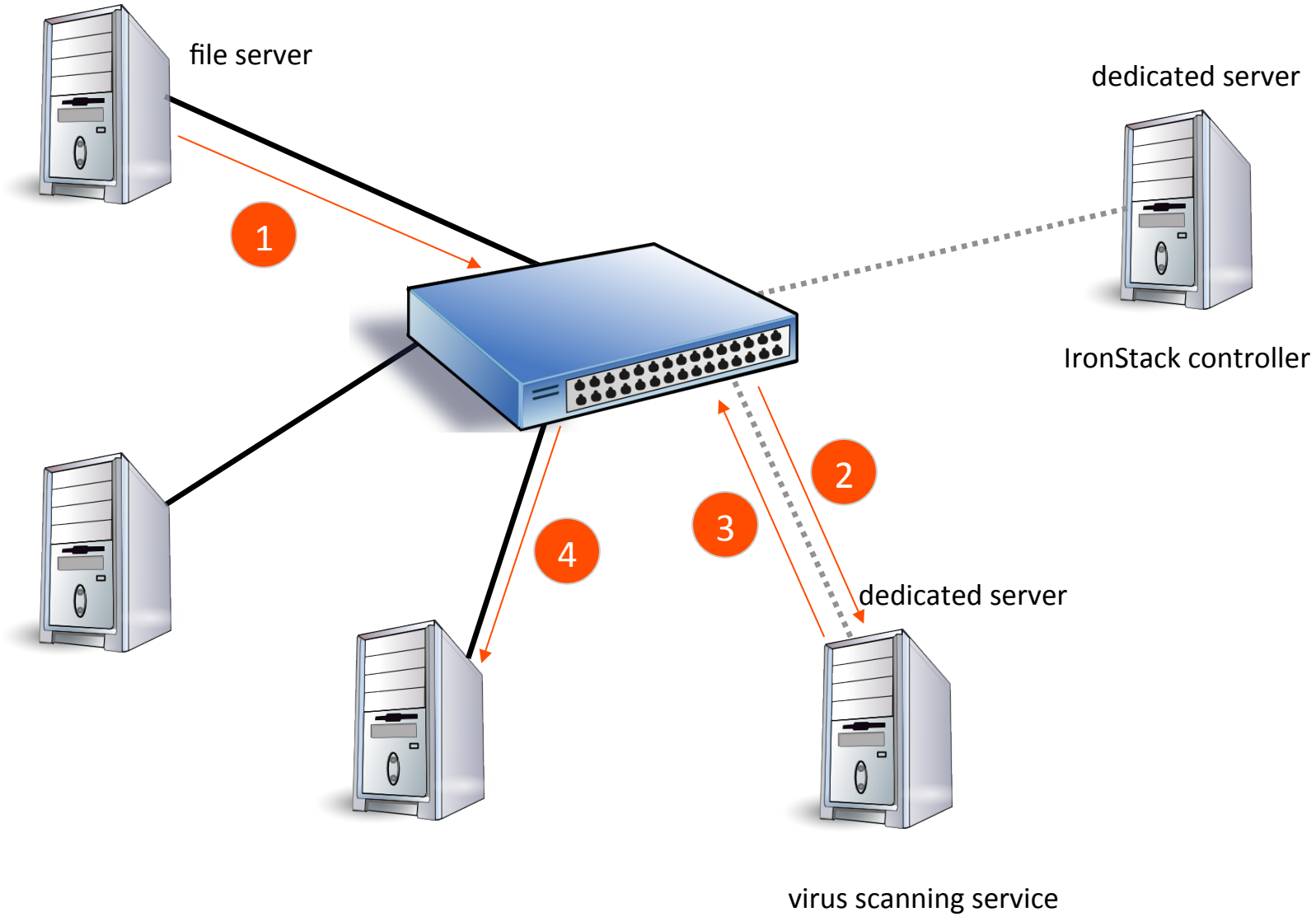
- TCP port 21 is for FTP.

- Special switch port connects to a dedicated server for virus scanning.

- Forward packets back from server to endpoint if file is free of virus.

- Can be load balanced for scalability.

IronGuard


file server
dedicated server
1
IronStack controller
2
3
4
dedicated server
virus scanning service

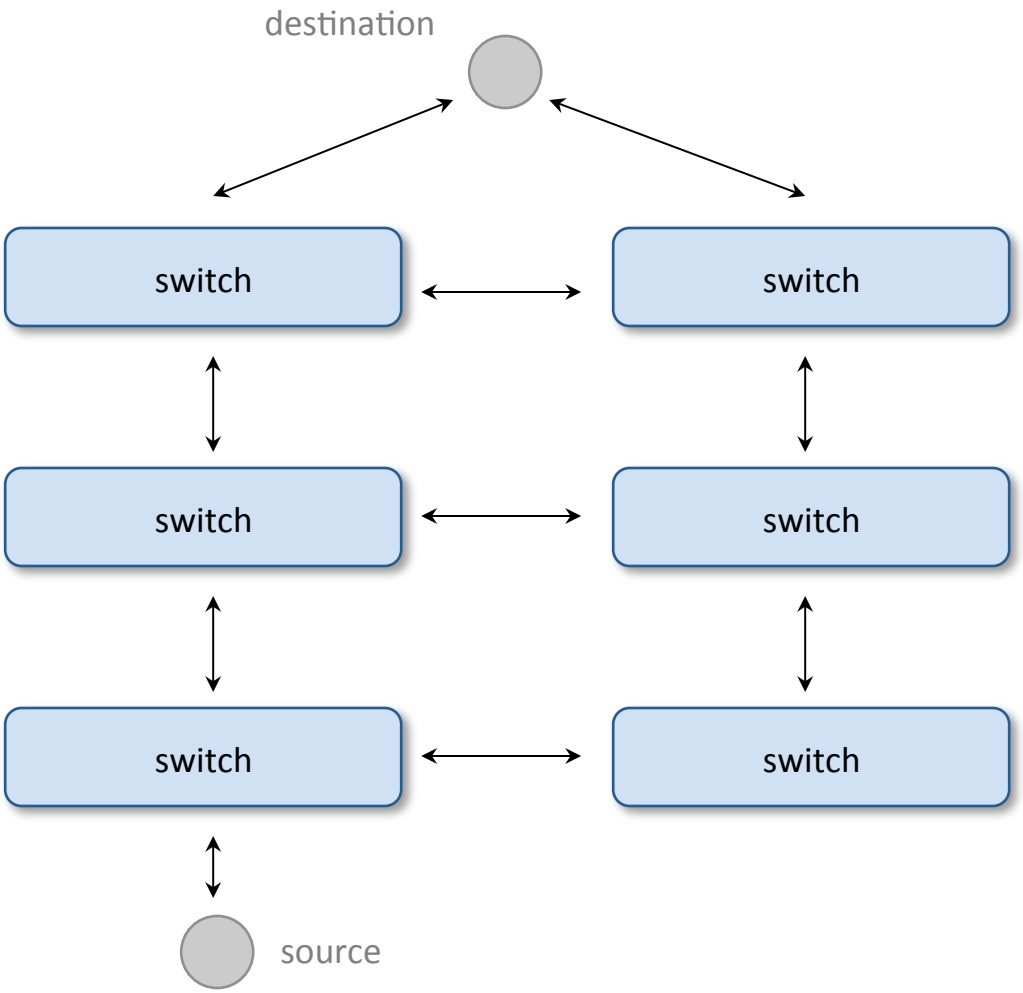
IronGuard

- In the context of embedded distributed systems, can be used to detect and stop proliferation of dangerous data.

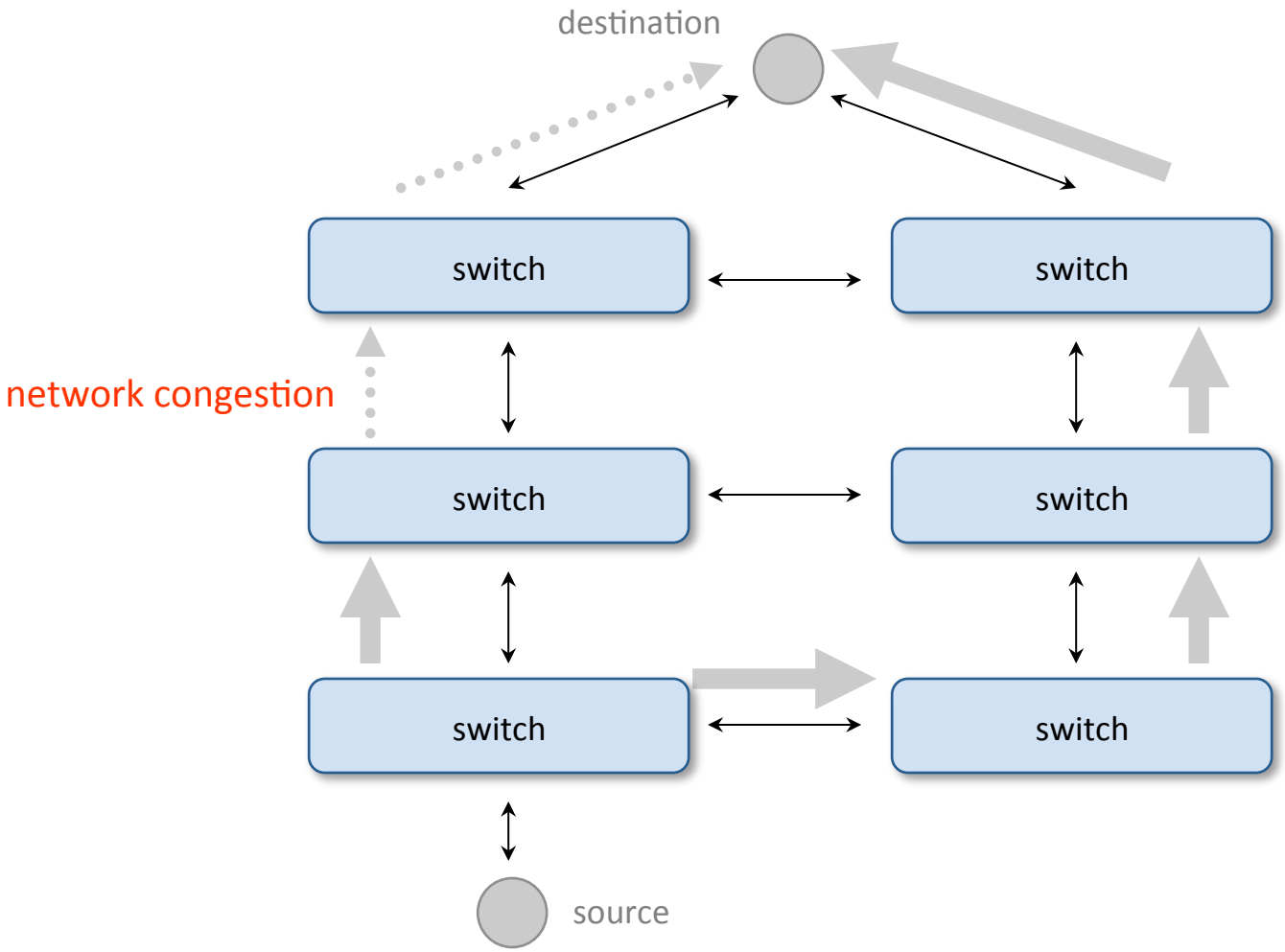
- Power grid: don't allow erroneous sensor readings to destabilize the grid.

- Telesurgery: prevent upstream attacker from using FIN attacks to close a connection.

Multipath routing



Multipath routing



Advantages of multipath routing

- Lower latency (first of any replicated packet is the one that counts).

- Greater perceived latency stability (less spikes; spikes less severe).

- Higher bandwidth (trade latency for increased bandwidth from extra paths).

- Bonus question: how to blend latency/bandwidth tradeoff?

Disadvantages of multipath routing

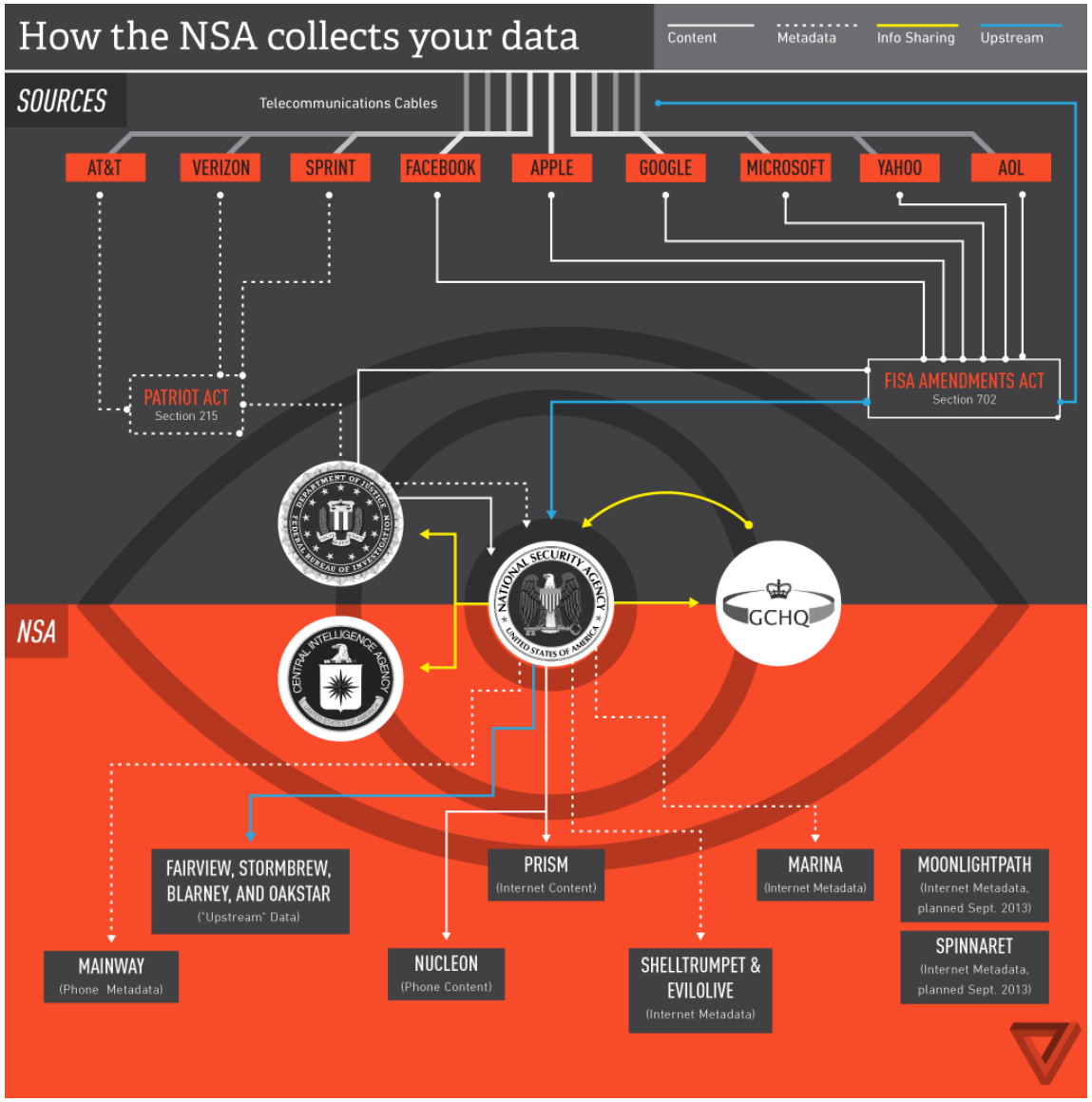
- More processing required at endpoints.

- Some overhead in controller-to-controller negotiations.

- Needs kernel support. (Aside: maybe not for TCP, but why?)

- May not work for all protocols (eg. UDP).

Physical data security



The blue line is a concern we may be able to act on.

Courtesy of TheVerge.com

http://www.theverge.com/2013/7/17/4517480/nsa-spying-prism-surveillance-cheat-sheet

Physical data security

Each of those little packets contains the routing information that allows the message to be reassembled and delivered. That's metadata. It performs exactly the same function as the envelope does to a letter.

That is why the metadata is so important and why you should be very concerned that the government is collecting it on every single American.

The final piece of this puzzle is how the government collects the data.

Knowing how the packets get from point A to point B isn't important, unless you intend to intercept it along the way. All of your communications go through gateways. Think of them as the onramps to the information highway. By standing at the onramps you can monitor the traffic flow.

Those onramps are your internet service providers—known as ISPs for short. They're Microsoft for Hotmail and Google for Gmail, AT&T, AOL and many others. Using the Patriot Act as justification, the government goes to the ISPs asking for information.

More than just getting historical data, however, the government has actually installed sensors at some ISP locations so that they can gather real-time information around the clock.

But wait, you might say: that's a lot if information. The government can't possibly collect and sift through it all.

Yes, they can.

The content behind the metadata sits in huge computer databases. Using keyword searches, as Edward Snowden described, they can get to any information they have pretty quickly. They no longer have to capture and analyze the data as it comes in. They have it recorded and ready whenever they need it.

Government collection of information about you started with your communications but it doesn't end there.

Courtesy of The Washington Times Community

http://communities.washingtontimes.com/neighborhood/red-pill-blue-bill/2013/aug/25/metadata-matters-ongoing-nsa-scandal/

Physical data security

- Why not take advantage of multiple paths to foil signals intelligence?

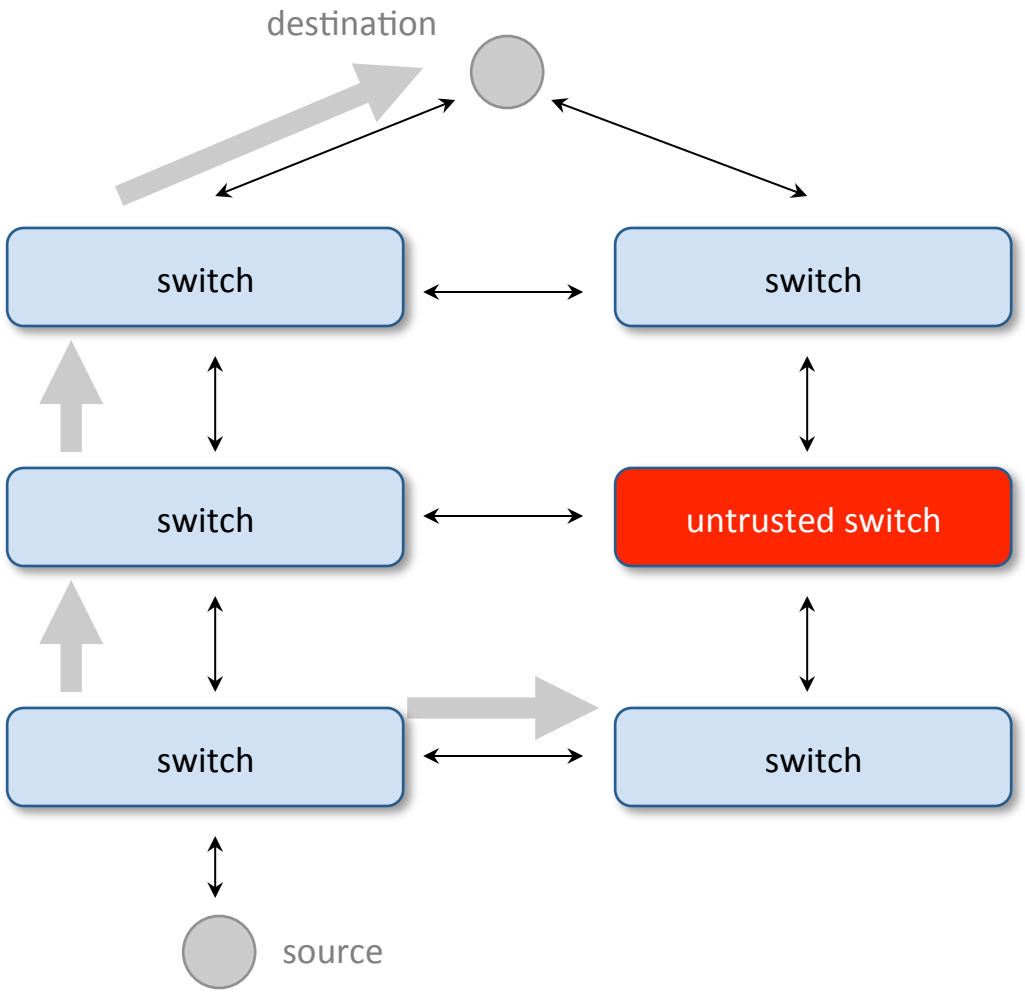
- Approach 1: blacklist known ‘unsafe’ nodes, don’t route over them.

- Approach 2: use only whitelisted nodes known to be safe.

- Approach 3: randomly send packets over different routes (trivial example: odd packets along link 1, even among link 2)

- Can combine with onion skin encryption for further guarantees.

Foiling signals intelligence



Physical data security

- Physical data security as described does not guarantee anything.

- Depends on controller to be trustworthy and non-byzantine.

- Does this mean it's entirely pointless?

Onion skin encryption

- Assume local controller has all the necessary public keys of every pertinent switch (really it's the controller tied to that switch).

- Since we know which routes to physically use, encrypt data (and the proposed route) multiple times, one for each hop.

- At each receiving hop, decrypt the data and forward it on iff it came through the expected switch port.

- Order of encryption should be the reverse order of traversal.

- Problems?

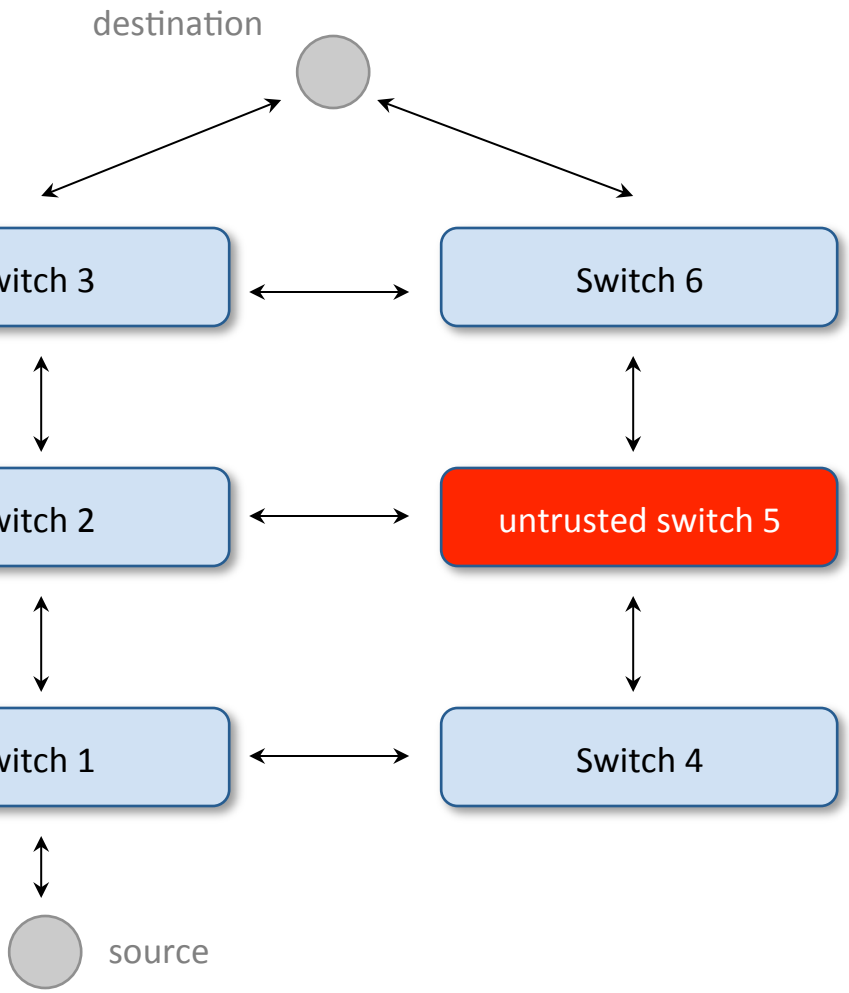
Onion skin encryption

E = encrypt
D = decrypt

D(E(data, k3), pk3) = data

D(E(E(data, k3), k2), pk2)
= E(data, k3)

E(E(data, k3), k2)



Problematic assumptions being relied upon

- You have to trust that byzantine switches/controllers are rare.

- Multipath actually exists.

- Realistic assumptions?

Not discussed: other IronStack capabilities

- IP/ethernet address spoofing detection.

- ARP spoofing detection.

- detection of CAM overflow attacks.

- disabling rogue DHCP servers.

- Will not be discussed during this presentation (not completely new ideas)

- Ask me in person if you are interested!

Questions?

IRONSTACK

NETWORK SECURITY FOR DEDICATED SYSTEMS USING OPENFLOW

Increasingly large number of critical infrastructure and enterprises employ dedicated, mission-critical networks. Recent waves of attacks against these systems motivate a closer look at the security of networks.

SOME PROBLEMS IN NETWORKING

Traditional networks have difficulty defending against spoofing attacks. Part of the problem is that it is difficult to tell genuine transmissions apart from bogus ones.

MAC SPOOFING

1. Alice connects to the network and passively listens for MAC addresses.
2. Bob connects and through DHCP or ARP requests, makes its MAC known to Alice.
3. Alice pretends to be Bob by spoofing Bob's MAC address. The switch updates port mappings so the MAC now refers to Alice's port. Alice can also launch man-in-the-middle attacks.

ARP SPOOFING

1. Bob broadcasts an ARP request for Charlie.
2. Alice intercepts this and replies to Bob, pretending to be Charlie.
3. Bob talks to Alice, thinking she is Charlie. Meanwhile, Alice is free to launch a man-in-the-middle attack.

CAM OVERFLOW ATTACK

1. Alice sends a barrage of packets with bogus MAC addresses to the local switch.
2. CAM tables in the switch overflows; this causes some or all legitimate traffic to the switch to be broadcasted out all switch ports.
3. Alice can now snoop on other traffic and launch other attacks.

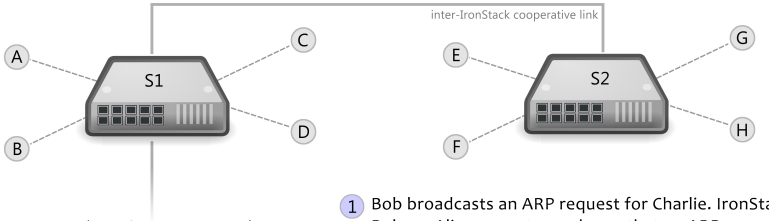
DHCP DENIAL OF SERVICE ATTACK

1. Alice issues many requests to the DHCP server for an IP address, each request potentially using a different MAC address.
2. DHCP server exhausts its pool of available addresses.
3. Legitimate users can no longer acquire IP addresses and thus cannot use IP-based network services.

SYSTEM ARCHITECTURE

IronStack is a hybrid software/hardware solution that mediates potentially unsafe operations such as ARP and DHCP. The switch controller operates on the L2/L3 layer and intercepts all incoming/outgoing ARP and DHCP packets for processing. If local processing is possible, this is done. Otherwise the controller acts on behalf of the requesting client to ensure safety. Hence, the controller insulates the client from potentially malicious attacks from other clients. To attain performance, non-ARP and DHCP packets are handled in hardware, without intervention from the IronStack controller.

EXAMPLES OF MITIGATED ATTACK



CAM entries (MAC/port mappings)

00:aa:aa:aa:aa:aa	port 1
00:bb:bb:bb:bb:bb	port 2
00:cc:cc:cc:cc:cc	port 3
00:dd:dd:dd:dd:dd	port 4

1. Bob broadcasts an ARP request for Charlie. IronStack traps the ARP request and replies directly to Bob, so Alice cannot even know that an ARP request was sent by Bob.
2. Alice pretends to be Bob by using his MAC address. IronStack detects this and notices a discrepancy between Alice's messages' MAC and port mapping. Alice is disconnected from the switch.
3. Alice attempts to make multiple DHCP requests for IP addresses. IronStack detects this and disconnects Alice from the switch.

DISTRIBUTED OPERATION

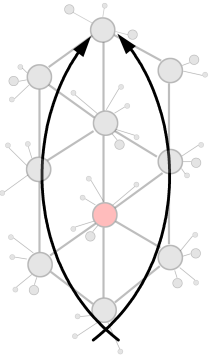
IronStack is designed to work enhanced with other IronStack-controlled switches, while preserving operation with regular switching hardware. Enhanced cooperative control allows negotiation of special features such as multipath routing, data striping and physical route security.

MULTIPATH ROUTING

route data along multiple paths for greater reliability and bandwidth.

PHYSICAL ROUTE SECURITY

specify explicit path from source to destination, or only via whitelisted nodes.



blacklisted node

APPLICATIONS

mission-critical, dedicated enterprise networks

WATCH THIS SPACE

Our full paper will present benchmarks about IronStack's resilience under duress against the above-mentioned attacks, as well as detailing its performance under normal operation.

PEOPLE


zhiyuan teo


ken birman


robbert van renesse

cornell university

If this sort of research interests you…

We're looking for students!

Please e-mail zteo@cs.cornell.edu