



Advance Computer Networks
Instructor Doctor Sameer Kulkarni
Department of Computer Science and Engineering
Indian Institute of Technology Gandhinagar
Lecture 55
Network Telemetry

(Refer Slide Time: 00:17)

NETWORK TELEMETRY

- Lack of fine-grained visibility into Networks has been a significant pain point for Network Operators
- Debugging correctness and performance issues in a live network is challenging
 - Packets not going where they're supposed to
 - Packets experiencing significant queueing or drops
- Measuring networks effectively requires collecting a lot of data
 - Recall: the number of packets every second even on a single 10 Gbit/s network >> Millions.
- Network telemetry is a technology for gaining network insight and facilitating efficient and automated network management.
- Network Telemetry Framework ([IETF RFC 9232](#))
- In-band Network Telemetry ([INT](#))





Data Center NetworksAdvanced Computer Networks

Let us try to look at another important aspect in networks called the Network Telemetry. This is about getting the visibility of what is going on in the networks. And this has been a significant pain point for network operators for several years. This is primarily due to the lack of necessary tools, the means to look at what is the aspects that is going on in the network, and what is the real time statistics at each of the networking devices.

All of these have been really challenging and it made debugging the networks and troubleshooting in case of any issues a very hard task. And this is where we need to relook and see what can be done better as we spoke about the network's softwarization aspects and carry forward into this aspect of network telemetry.

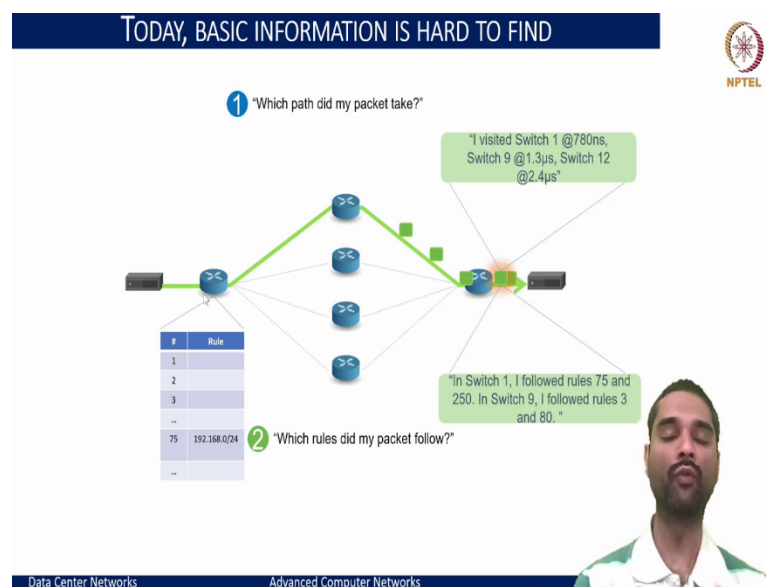
And this is all about measuring the networks effectively so that we are able to collect the data, but beware like we spoke about the several gigabits links, and even when we spoke consider just 10-gigabit links, we are talking about millions of packets per second, and now with hundreds and 400-gigabit links, this will be multi-fold.

So, anything that we want to define are provided the mechanisms should be able to scale and be able to operate at such a very high traffic loads and that becomes an important aspect as well. And how we define network telemetry is basically a technology through which we can gain insights into the network and facilitate for efficient management.

And in this context, several of the works were put forward in over the last years few years. And the one that came from the IETF which was very recently published in May 2022 is this IETF RFC 9232, that is, bringing and defining or charting out what the network telemetry framework should be, so that we are able to do these tasks efficiently.

And the other series that happened with the networks softwarization group of what we looked at before and how we try to go towards the programmable networks is the In-band Network Telemetry for data plane network analysis, and we will try to look into INT and also see how this has resonated into the network telemetry frameworks in specific aspects.

(Refer Slide Time: 02:41)



Primarily to just think of the background if we had a topology as shown here a very simple one, with several of the routers just between the two end hosts where the network would be exchanging information. Now, the important aspect that we would want to consider when any of the packets get lost in the network is to know where exactly or which router in the path did it get dropped.

And if we are seeing for latency-sensitive flows that we are missing the deadline, we also want to know at which router in this topology resulted in having a lot of queuing delay or which contributed towards the majority of the delay which made the deadlines to be missed and if they were throughput sensitive we want to know which of the links are currently constrained or over-utilizing the bandwidth and affecting the flow's throughput.

So, all of these questions that will come when we look at such a simple topology, and even to the very basics when the packets go from source to the destination, we would want to know what exactly happened to that packet, and we can see when we spoke of data center networks, there are a lot of paths through which the packets would flow.

So, it would also be important to know which paths did the packets traversed in the first place, because there may be a lots of packets that may be traversing through different paths in the network. So, as they traverse, you will also want to know exactly the list of the switches and routers that they visited to eventually make it to the destination.

So, that means they should be able to provide us saying the packets visited basically switch 1 at a particular time, and then going from switch 1 to switch 2, they visited and left the switch 2 at a particular time instance and eventually made their way out to the end host.

And likewise, when we see the packets flow through these different devices, we also want to know like if switch 1 has visited, what kind of rules were applied to it in terms of the way that forward out on port X, port Y, port C or maybe do some processing on the headers like we say the NAT whatever translations that we may be doing.

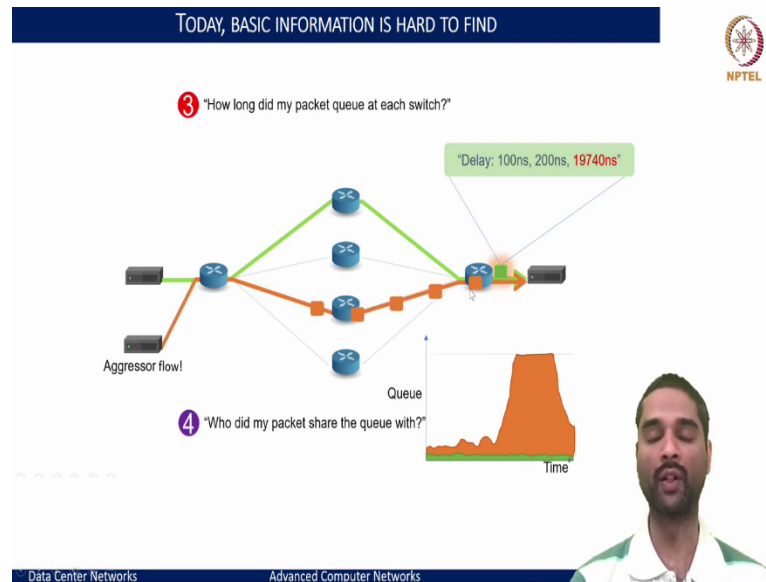
So, all of these information in terms of any transformations or any rules that each of these devices supplied will be an important aspect also to look at to say that as the packets went through this router 1, they were applied with a specific rule to say, they match on 192.168.0.24 with a match rule where they would process certain actions.

So what kind of actions, how many actions were applied at each of the switches, as the packets traverse through them would be an important aspect because many times if we have a miss, properly configured NAT or a firewall rule, we may not have the connections to get established and it may be very hard to debug the issues were at those particular devices where the firewalls

were dropping the packets, where the NATS had improperly configured the headers and not making the packets to go through.

So, there can be various of these aspects that we will also learn if we knew what rules were applied at each of these switches and routers.

(Refer Slide Time: 05:58)



And likewise, as the packets traverse, we will want to know at each of the switches or routers, how much time did they get enqueued because if we are seeing latency-sensitive flows, we would want to know which of the routers or switches contributed towards the majority of the latency and for what reasons.

So, if we are able to extract information as saying, I encountered the delays of 100 nanoseconds, 200 nanoseconds, and like this 19740 nanoseconds to each of the routers, then we can very easily pinpoint and say that the main concern seems to be in the router 3 where it is experiencing abnormally high delay.

And likewise, we will be able to know if we are able to capture this information for every packet in the flow; we will be able to see over time how the queues have built up at each of the routers, what is going on between each of the routers in a very fine-grained manner. And this would also

help us to know if there is a latency that has been spiked at this router 3. But this could be due to some other externalities.


And we would want to know what kind of externalities caused this kind of an effect. And in essence, that would mean that maybe at the time, when this packet was reaching router 3 and having the packet go through this router, maybe there were other flows, that were hogging the bandwidth, and maybe they would have occupied the queue.

And because of which, the green packet ends up having to be killed for a long time suffering abnormally high delay. So, there can be several of the aggressor flows that may be flowing through the switch at the same time. So, you would want to know, if the packets shared the queue with what all kinds of the flows at a particular device.


So, if we have such an information, then we would be able to make the deduction that the green packets lead to a very high delay or latency queuing delay, that is because it is just router 3 was occupied with lots of packets from this orange path or orange packets. And these are the concerns for seeing this abnormally high delay.

(Refer Slide Time: 08:06)

TODAY, BASIC INFORMATION IS HARD TO FIND




- 1 Which path did my packet take?"
- 2 Which rules did my packet follow?"
- 3 How long did it queue at each switch?"
- 4 Who did it share the queues with?"



With P4 + INT we can answer all four questions for the first time
At full line rate. Without generating additional packets.

Data Center Networks

Advanced Computer Networks



So, in essence, when we want to build this network telemetry framework, we would want to understand and answer all of these questions like which path did my packet take? Which rules

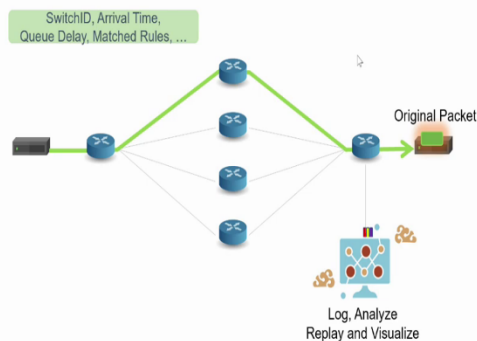
did my packet follow? And at each of the switches, how much did they spend time in getting queued before getting processed? And if they were queued long enough, we would also want to know with whom all did they shared the queues while they were residing at the queue at a given router. So, answering all these questions, was in effect. The goal for us is to do better network telemetry. And this is where the P4 group tried to come up with this In-band Network Telemetry, which would try to answer all of these questions.

And most important part here is not just to answer these questions, but also to ensure that we are able to do that without impacting the packet processing at a line rate. That means we should be able to do that for several million packets per second in a non-intrusive manner. Or it could be means where we could develop mechanisms, which can allow us to do that in a very least overhead fashion.

And it is also important that if we have poor packet details, we do not want to be generating extra packets into the network, adding to the bandwidth tax. We also want to minimize this bandwidth tax that would pay for just collecting this information. So, think of if I have one million packets, I add one more million packets, making overall two million packets to be traversed to get to this information that would be infeasible.

So, we would want some intelligent and nice mechanisms that would allow us to have this done in a very low overhead fashion.

(Refer Slide Time: 09:45)



In a nutshell, In-band Telemetry tries to leverage the mechanism whereas the packets traverse through each of the switches and routers, which we now consider to be P4 programmable, would be able to look what is the current characteristics and the status of each of these devices and add the information into the packet header as the packets traverse for the required details that we want to collect and log at each of these routers.

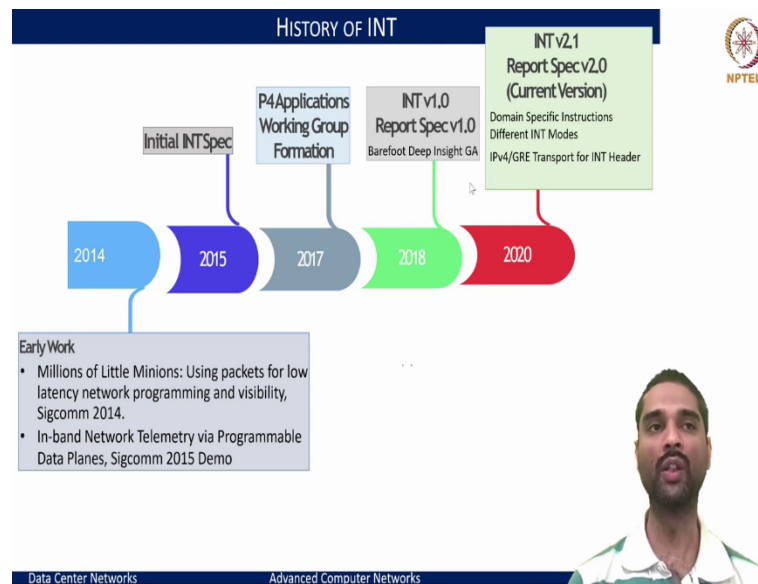
So, as the packet reaches router 1, router 1 would then stamp the details in terms of what is the switch ID, what was the arrival time of the packet, how much time did that packet spent at the queue within that switch, and what were the matched rules on this particular switch, all of these information can be padded on to the packet and then forwarded it on to the next router.

And as it reaches the next router again, the next router would do the same, adding its set of information similarly and appending it to the packet and likewise as the packet traverses through the set of switches and routers, we will see that each of those routers would append their end of the information.

And as it reaches the last hop router, the last hop router can take out this information and provide these specific details that were collected as the packet traversed through this network to provide for further logging and analysis of the details while sending the original packet to the end host as desired.

So, in essence, we make the information to be made available In-band within the packet as the packet flows through the network and extract out the information at the last hop, and ensure that the packets that are delivered to the end hosts are the same as what the source transmitted.

(Refer Slide Time: 11:38)



So, let us try to understand briefly about how the INT evolved and what are the key modes of operation that INT brings forth for the network telemetry. So, the early work started in around 2014 with a paper Millions of Little Minions in Sigcomm 2014, which was followed up by In-band Network Telemetry via Programmable Data Planes as a Demo paper in this Sigcomm 2015.

This originated the idea of how in-band network telemetry can be used and the development of a framework that will allow for collection and reporting of the network state in real-time at the data plane without requiring any of the interventions or the need for the control plane as such. And this work initially resulted in the culmination of an initial INTSpec in early 2015.

And it started taking shape with respect to saying what should be the means to facilitate design to information wherein the packet header if the information has to be embedded, what should that be and what kind of information that needs to be extracted, and what are the means to specify which information needs to be extracted at a switch and where to embed. All of these started to take shape.

And a P4 applications working group was formed in 2017 to bring about the exact means of making these prototypes to work. And this led eventually to the formation of the version 1.0 of the INTSpec, with the development were led primarily from the Barefoot, which were the pioneers in the P4, and several other board members started to join and operate in building towards design INT v1.0 spec, and this evolved over the recent years.

And there have been various adoptions that have been done from Cisco, Arista, Xilinx, Intel ONF, all of these people have played a crucial role in paying towards the newer variants of the spec. And the most latest version is INT v2.1, which is the current version of the spec, which defines specifically the what are the Domain Specific instructions that a switch or a router should be operating and what are the modes of operation that INT should be facilitating.


And if this information has to be embedded within the packets and sent, what is the means to tunnel the headers so that the original information is not affected? While the switches should be able to understand the INT instructions and do specific processing.

So, all of these details have being laid out in the INT v2.1 spec, and in a brief time, let us try to understand some aspects of how this INT processing happens and what we mean by different modes of operations, what are the key tenets and principles on which the INT designs were considered and laid out?


(Refer Slide Time: 14:33)

INT DEVELOPMENT PRINCIPLES

- Leverage **Programmable** P4-based Data Plane
- Release Features and Capabilities at Software Development Velocity
- Applications / End Points completely agnostic of INT underneath
- Flexibility is of utmost importance for rapid innovation and minimum barriers to adoption:
 - Flexibility in Modes of Operation
 - Flexibility in INT Header Location: INT over TCP/UDP, VXLAN/Geneve, IPv4 GRE
 - Flexibility in Instruction Definition: Domain Specific Instructions
 - Flexible Report Format Definition
 - Flexible Metadata Semantics: YANG model based reporting of metadata.



Data Center NetworksAdvanced Computer Networks



So, the key development principles were primarily to leverage P4 programmable data plane aspects, so that the switches and routers could embed and look for necessary information, add that data, and extract that data in a defined manner and because now we are seeing the programmable data plane it also gives us the flexibility to define the header formats, the way we would want.

So the main insight here is to ensure that we are not confined to any specific protocol version or packet types; we could be generic enough to facilitate any of the data types. And because it is now programmable, that also means that we will have the software development at a very high velocity. So, the key development principle also involved saying that the hosts have to be agnostic to the INT information that will be generated underneath in the network so that the endpoints need not have to worry about how this information is being added and how this information is being collected and analyzed and work as if there is no change. And in terms of the network, when these switches and routers that would want to append and operate on this information, they have to be flexible in terms of what kind of an operational mode they would want to work where the data would come because we have seen that if there were IPv4, IPv6 routers, you have a tunnel.

And if there is a LAN, and you have the VLAN, headers, VXLAN header, so all different kinds of tunnels, different kinds of packets would flow in the network. So, we do not want to be binding to any specific protocol headers for this INT information. And INT should be facilitated over either the TCP or UDP flows with VXLAN or a Geneve encapsulation or IPv4, GRE all of these regardless of what kind of encapsulation or packets they are, we should be able to facilitate supporting INT.

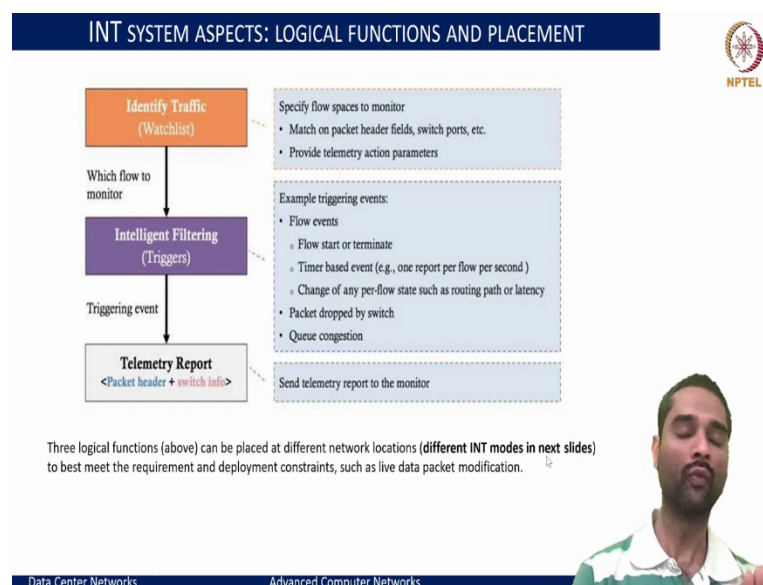
And this also means that, the packets we will want to process them and add the information, the instructions that define what kind of operation or what kind of data that we just need to add, should also be having specific instructions. And we want to basically define domain-specific instructions and at the same time, make sure that they are flexible enough so that they can be expanded for the needs that the network operators would have.

In terms of reporting the information and what model you want in terms of facilitating the data to go through different switches and update to the monitors, we would also want to have flexibility there in terms of what can be supported to say that we want to have a mechanism where the network packets can be processed independently.

And each packet should also have this metadata semantics, which tells what kind of information is embedded within each packet and so, that it can be processed independently and sent for the log analytics. And here, it tries to take advantage of a YANG data modeling, which is basically the modeling which defines the hierarchical data structures that can be used for operations based on the network configuration management or the netconf.

And these provide the tools for the network administrator to enable automation of the configuration across heterogeneous devices. And INT tries to leverage the same concepts in trying to deliver the network data.

(Refer Slide Time: 18:02)



And if we look at what the INT then consists of, it is basically three key aspects. One, if we want to monitor, we should be able to identify the traffic that we would want to monitor. And in essence, what the INT specifications define is a Watchlist. And this Watchlist consists of the kinds of flows that we would want to monitor that is to specify the set of flow spaces which are of interest for which we would want to collect certain data.

And when you want to collect certain data, we would also want to say what kind of instructions that you would want to apply like do I want to collect the delay parameters, do I want to collect the queuing parameters, all of these information that we want to collect should also be able to be defined.

And in this essence, the Watchlist tells basically the flow space which could be IP five-tuple or any additional packet header fields, not just confined to the five tuple if we want to match. And once we have defined the Watchlist and what kind of an operation that we want to do.

The second important thing is to also say once you have a match what kinds of events that you would want to generate or trigger that you would want to set so that the logs can enable you to look out for specific actions for example, dropping off a packet as a kind of an event. Or if you see that certain packet from a particular source IP on a source destination port is coming, that is a kind of an event.

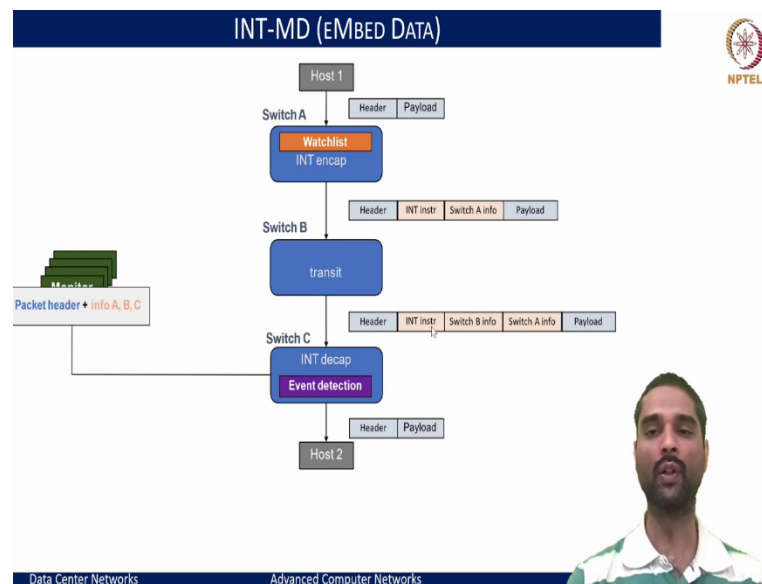
So, we should have the mechanisms to say what kind of events are generated because of these packets, which again through P4, we can program the switch to say when certain conditions are met and what to do when such conditions are met. And this is where the intelligent filtering component comes in to say that you want to be looking out for the flow events.

And if we are talking of just the TCP whenever the TCP flow starts like with a sync flag, that is a very simple example to say that we start to see a new TCP flow at this point or when we have the TCP flow terminate with FIN ACK flags, then we can also mark these special events. Or even if we want to have any per flow statistics that on a routine that we want to look for, or a queue congestion, or a new condition.

All of these basically become the triggers on which we would want to set, that means we need a mechanism from the INTs, to say how to watch out for these as a key of instructions and let the switches or routers look out for these and process them. So, the means of a program that will do this, and instructions that will specify what program needs to be run on a switch.

And once we collect all this information, we need the means to report to a certain entity and the means to formulate the report and send it to an agent. And all of these aspects are the core tenets of INT; let us try to cover some of the INT modes in terms of how the operations are being defined, and what information are being prepared at each of the switches, and how the information gets logged on to the monitors.

(Refer Slide Time: 21:13)



So, the INT comes with various modes of operation. And one of the very simple modes is the INT EMBED DATA or INT MD mode. And the construct here is to say that as we have the packets go from one host to the other end host, they will traverse through different switches. And for each of the switches, you would configure the Watchlist.

And once the packet flow space matches within that Watchlist, then a certain kind of actions that need to be taken as instructions and specific data associated with those instructions would then be padded for each of these flows. To think of host 1, which is agnostic to the existence of this INT, would send the header and payload information to make sure that data reaches to the host 2.

And as it reaches to switch A, switch A would look up in its Watchlist and identify whether this header matches any of the entries in the Watchlist. And if there is a match, and it will then pad what are the corresponding instructions that it needs to follow in terms of what we want to monitor, what are the events that we are looking out for this specific kind of a packet and then

add that information in terms of saying now we want to have these packets to be processed with these kinds of instructions.

And for these current instructions, what is the information that you have at a switch A so that gets padded as basically the INT instruction header and the info switch A, which actually processed this information and padded? And in this terminology, switch A is basically the ingress switch at which the INT monitoring starts.

And any subsequent switch that the packets would then go through will basically look at this INT instruction header within the packet and then start processing and updating the information. So, any subsequent switches that you would see what we call as the INT transit switches, while the first switch where this information gets added is what we call INT encapsulated or the ingress switch.

And the last switch where this information would be taken out is called INT decapsulated or the egress switch. So, as we see that the packet goes through the switch B, switch B would look up this INT instructions and add its kind of information maybe the delay if we are trying to monitor for what kind of flows were in the queue at the given point as this packet progressed, all of that info would then be padded on to this packet and forwarded further even including if we are monitoring for the kind of rules that were applied at each of the switches. All of this info will be written out into switch B info and then forwarded over to the switch C.

And as it reaches the switch C, switch C would also add its information with respect to whatever INT instructions were specified. But before forwarding the packet on to the host 2, it would decapsulate this INT information, extract it out, and then just transmit the header and payload information to the host 2, while sending the entire INT information that is collected as a part of INT instructions switch A info, switch B info and switch C's own info all of this gets added and sent out to the monitor.

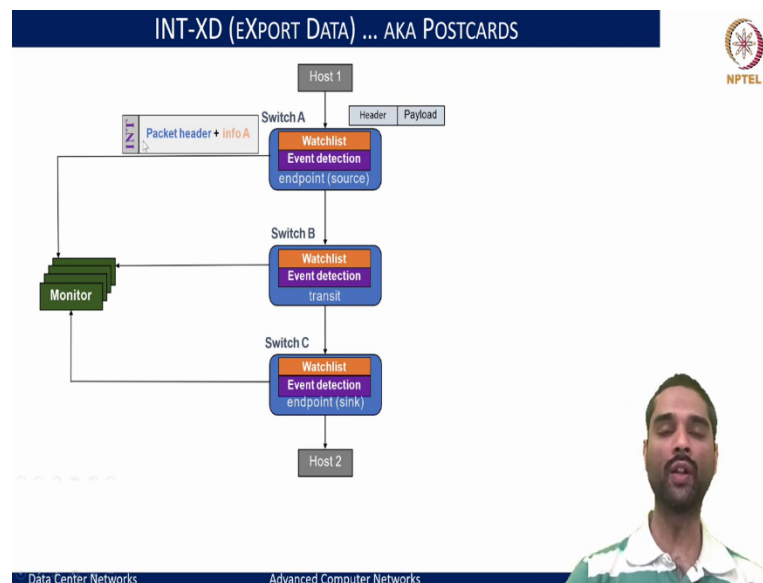
So, this is a very native means to say that we are able to generate the INT information within the packet itself and that is why it is called the EMBED data mode. And with this model, the information would be extracted out at the last egress switch and sent to the monitor while the packets between the end hosts go as plain.

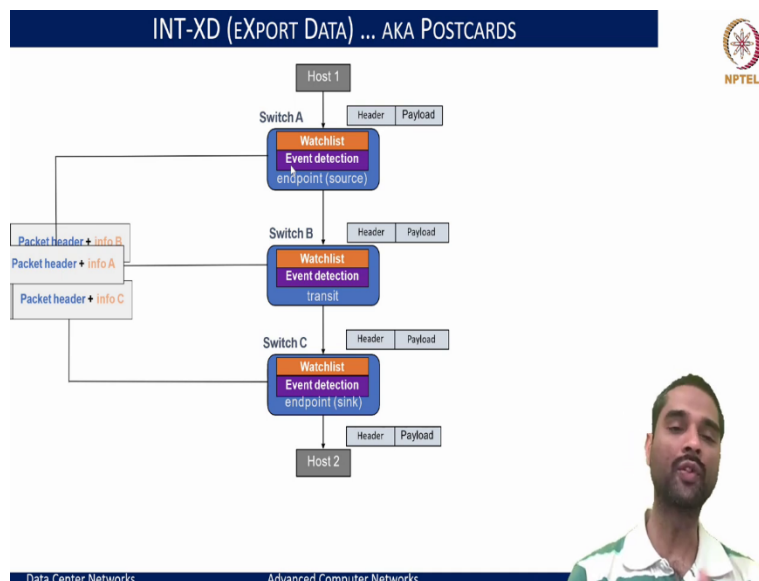
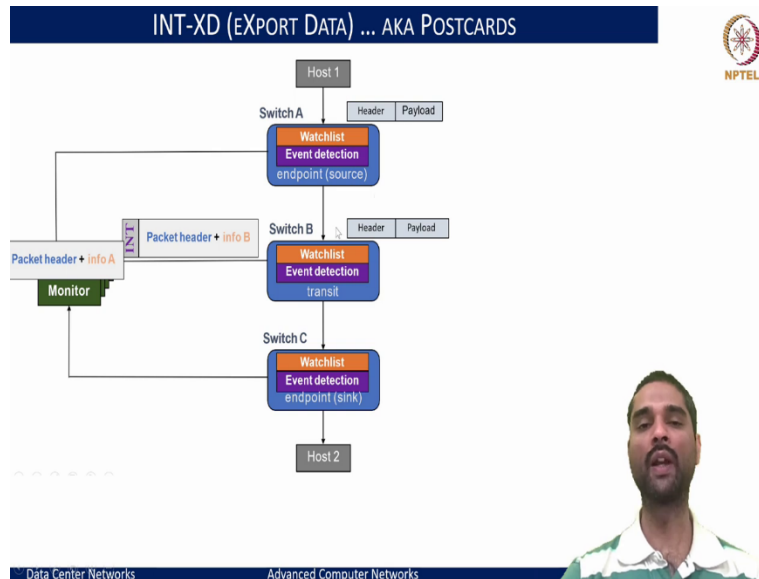
In this model, what we see now is that the information that is being added for the INT makes the packet to bloat as it traverses multiple of the switches. And here it could be possible that the packets that need to be fragmented due to the link constraints that you may have in terms of the MTU may have some concerns.

Nonetheless, this is a very classic simple format in which we are able to monitor within the packet, and here the switch A which has the Watchlist, is what we call as the ingress or the INT source, and switch C, which is the last switch that processes this INT is called as the decap or the sink switch that basically takes off this INT information and note that when we have lots of switches and lots of information that gets padded, it may become difficult to fit within the MTU size.

And also, the other thing to look at is that as the packet progresses, we are increasing the size of the packet that flows through each of the switch. So, that means the bandwidth constraints could start to increase if we have loads of switches through which the network has to pass through. And here, we also have to be cautious about what kind of information we can embed and how efficiently we can embed so that the headers or all the information that we pad is minimal.

(Refer Slide Time: 26:32)





The next kind of model is what we call as the EXPORT Data Model. And here the information that used to be padded at the source and then the one that used to be padded at each of these individual switches and then sent back to the sink, we can try to give it another model wherein if each of these switches have the information configured, they can do different kinds of event detections altogether.

Maybe I need certain kinds of information at switch A, while I would need certain other kinds of information at switch B, then I want the configuration flexibility to define what gets matched at A through a distinct Watchlist. And what gets matched at B through a distinct Watchlist.

And for the same flow space, I may have different instructions for switch A, switch B, switch C and likewise, while in the earlier case, we saw that the packet would have the INT instruction which specifies exactly the domain-specific details that each switch has to carry. Here, we are making much more flexible to say that each switch may have its own domain-specific instructions for a given flow space.

And this way, we can have the collective information that can be collected across different devices. So, the mode of operation is like once the packets are sent to the first source switch, whichever the switch that sees the flow space of a host 1 packet to match in its Watchlist, what it would then do is generate the INT packet separately.

And based on whatever the Watchlist rules that are being configured like, if switch A was asked to report the kind of flow rules that it matched at this particular switch, consider this is a NAT then we will want to know what kind of flow which index did it match what were the changes that it did and the switch A will then accordingly prepare for each of these packets that it matches into the packet header and information taken as a separate INT packet.

And then the switch A will send this information to the monitor at the point that it processes this packet and forward the packet as it is to the switch B. So, you can notice now that there is no extra information that is padded to this packet. So, there is no INT header, the domain specific instructions nor the data that is being added to the packet goes as it is.

And likewise, when the switch B sees this information and if it has its own Watchlist where this header matches and it might have its own kind of information that it wants to extract like considering this is a core switch at which the information is flowing, we will want to know with what are the flows that this switch packets are competing with.

And we may want to extract this specific information and the delay information that this switch so that switch B can add whatever information that it wants to padd and craft out a separate INT packet and send it to the monitor while sending the original packet towards its destination as it is.

And likewise, again at the switch C it may have its own Watchlist. And if it were typically a sink, we would want to know what is the total delay that it might have observed as it is coming or any other metrics that will want to look for and extract that kind of an information and send it to the

monitor for the same packet and then send the packet to host 2. So, in this model, we have basically ensured that the packets are sent as it is across the path.

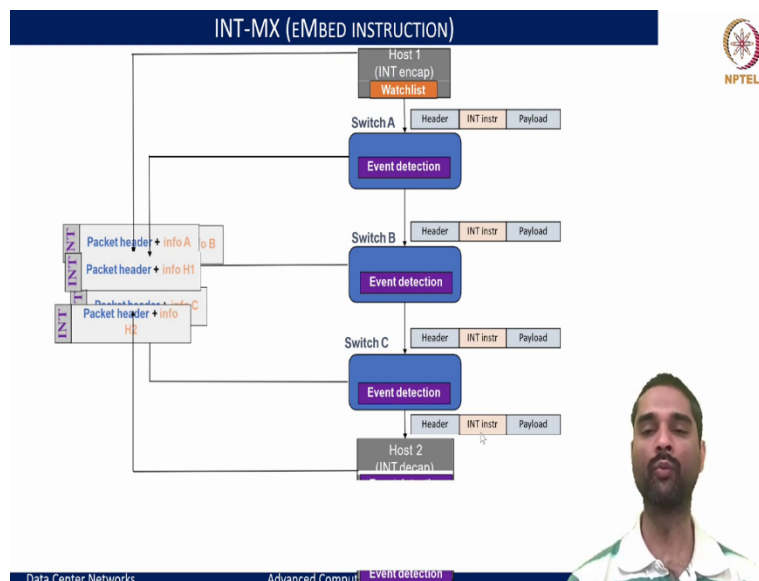
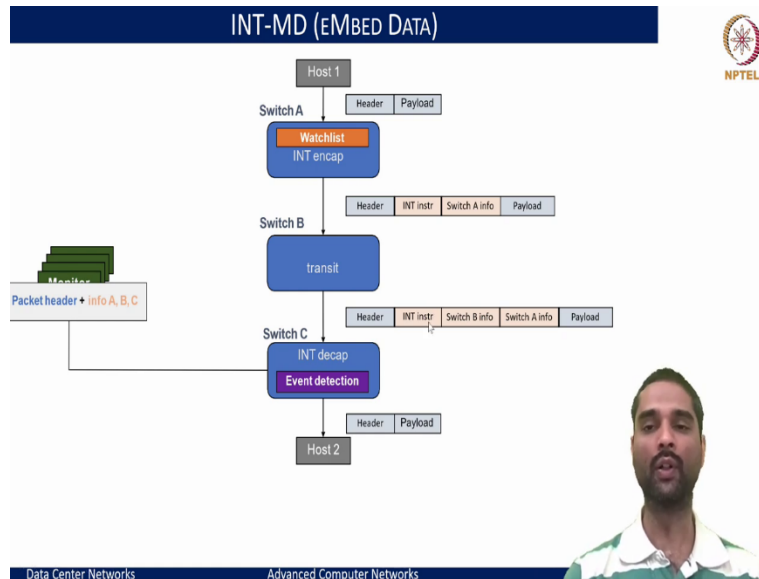
And two, we have provided the flexibility to say that different switches can operate with different Watchlists. Note what that means this if switch B had a Watchlist that did not have any match for this header, switch B would not be sending any information towards the monitor. So, we have given this flexibility now to say that we want specific points in the network where we want to observe for certain characteristics that can be done through this Watchlist configuration at each of the switches.

And further, we are making sure that the information goes in the off-band manner for the INT towards the monitor. So, each of these switches will be configured with a monitor where they will want to send and it is likely that we may not have just one monitor, we may also want to build this monitor at different points, which is more flexible and convenient to collect different kinds of information.

So, the monitor, think of it as an aggregator and a set of aggregators, one may want to look up for latency, one may want to look up for the rules, all of these information can be split and not necessarily that you want to do all of them at once. So that is the flexibility this INT EXPORT data or a POSTCARD method where the information of INT is embedded as a specific packet as a postcard and sent towards the monitor.

This in a way, we can see that it complements what we saw earlier with INT Embed data format where the data was flowing in-band. Here, the INT information that is collected is in a way flowing at each switch, although in-band with respect to how the packets are going to be processed at each switch. But the information itself is sent out of band in a separate postcard.

(Refer Slide Time: 31:51)



The alternative way to say it if we consider way that we just so you had to maintain the Watchlist at each of the switches. And when we say of Watchlist, we are speaking of millions of flows for which we want to keep this information. And this can be costly at times to maintain it at each of the switches.

So, the means of now to rethink if we had the Watchlist just at the host where we have abundant memory. And it would then see what packets before it is trying to send out to say what instructions for each of the switches to do. And then you would want the switches not to keep any memory for Watchlist because this also means that you have to match and then do some

processing. We want to eliminate those overheads in terms of processing cycles and memory overheads, we can do that as well push it to the hosts or even it could be not just the host, but it could also be one of the switches to say as an ingress which the first one where we could add this Watchlist. And once that is done, what that will follow is to say that you have domain-specific instructions that are added for a particular packet.

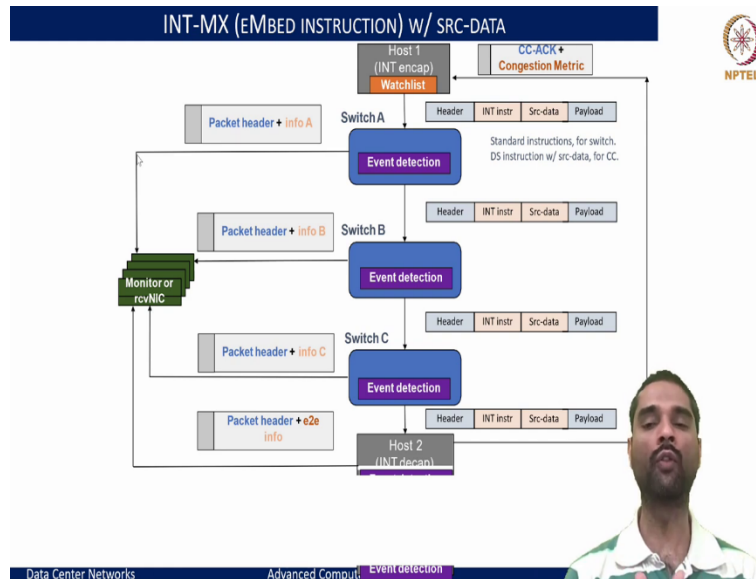
And as the packet traverses through different switches, you would want to detect just the events rather than looking up for the Watchlist in a specific memory and then match the instructions, you will look up the instructions right within the packet, and take necessary actions to do the processing and generate the events out. So let us see how this, in fact, operates.

So, as the packets go through the host one, it would generate the Watchlist, send the information at the same time, it would pad this INT instruction header. Now, this information is sufficient for subsequent of the switches to look up and generate their packet headers. So likewise, the postcard method, you can see that each of the switches will then send their part of the information towards the monitor.

But what to process on the need not how to keep a specific Watchlist but can take the queues from the INT instruction that is padded in the header and do the specific processing and send the information towards the monitor. And eventually, the packet reaches with just as it was sent out from host 1 with header INT instruction and payload, and then INT instruction can be consumed at the host 2 and then sent with just the header and payload information.

So, this EMBED instruction model helps to say that we have uniformity in terms of what each switch would operate on in terms of the instructions and the information that is being collected by each of the switches for different packets being sent in-band with respect to the packet processing towards the respective monitors. So, in a way it combines the best of both the worlds.

(Refer Slide Time: 34:41)



And we could also have a means where we could say why just have the in-band instruction? at times we would want to switch at the subsequent hops to know what happened before and if they need to have that intelligence. We need specific source data. Consider when we spoke about the delay where you are having a certain delay budget, and you want to work on within the delay budget, then I have to know what was the exact delay that it incurred at the prior hop.

So that I can do the processing in a different manner. So having not just instruction but also specific data could help the subsequent downstream switches to operate in a better manner. And also ensure that if there is some information that you would want, at the each of the ends to process, this would help to say that I can add minimal information in the in-band model of the packet as instruction and data.

And any other information that we would want to collect as a postcard model could also be sent. So, this is equivalent to saying we have the embed instruction with source data that we could add at each of the switches but confined to a subset of memory that we would want the packet headers to build up.

(Refer Slide Time: 35:58)

- Real-Time Fine-grained visibility is a game-changer and opens up a spectrum of use cases
 - Simplified Troubleshooting
 - Intelligent Path Selection: Choosing the best among Equal-Cost paths for optimal performance at flow / flowlet granularity
 - Congestion Control
 - Network Management and Capacity Planning → reduce OPEX



Overall, with these modes of operations that we discussed, INT opens up for a very nice mechanism for fine-grained visibility, and generation of the key events that would help us react much more quickly and have good troubleshooting aspects. So, it opens up for very simplified means to troubleshoot network issues wherein we can generate and add the INT instructions to meet what kind of characteristics we are looking aspects that we want the switches to append because it is programmable, we can even add any new sorts of information or programs so that the switches could add this information very readily.

And second, we saw about the congestion as important aspect that we want to meet in a sub-microsecond timescale. And if we have to do that, and we want to utilize the paths of the network in a much more better fashion, we want this intelligent path selection to base on the information that we gather in real-time.

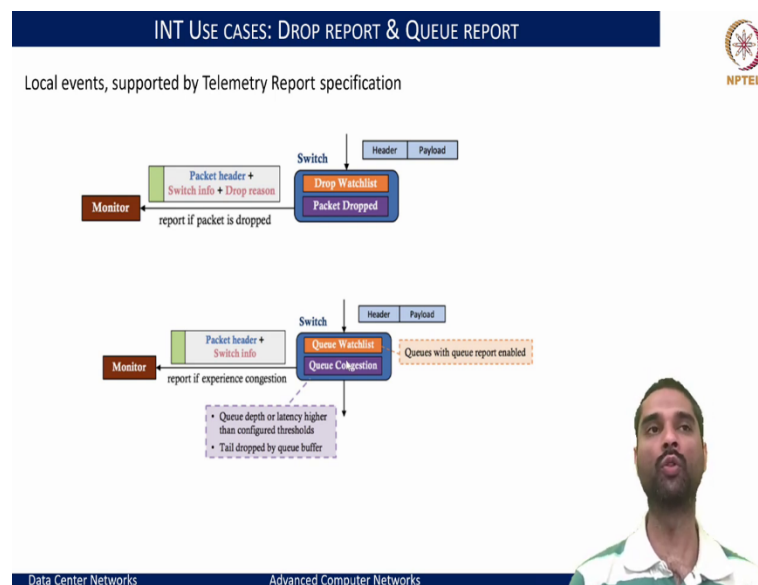
This is possible with INT now to check for which of the paths are having what congestion levels or where the queue is building up where the queues are low and accordingly make the decisions of routing the packets and there have been several works that have come like hula that try to use basically this information within the networks and to better optimal path or optimal routing.

And likewise, with respect to congestion control, and network management, and overall capacity planning, all of these can now be done much more easily. So, this would also mean that the expenses that we would have to do on the operations of the network will also be lowered because

now we are able to do it in an automatic fashion where the monitors that are going to be run as programs would be working in the background without needing much of the overheads from the network operators.

So, all of these in a way greatly simplify the network operator's life and also enable to provide rich visibility of what is happening in the network. And to add what we can call as automation, for course corrections or updates that we want to build on the network through the use of INT.

(Refer Slide Time: 38:17)



Let us try to look at just a couple of use cases in terms of how this INT has been very useful. So if we consider like a case where the packets get dropped in a network and we do not know where exactly are the packets that are getting dropped or if we know that the packets were dropped at a particular switch, but for what reasons will they drop is not sure.

Because there may be dropped due to the corruption of the packet headers, which the packet is not able to be read and be processed or it could be that the packets are dropped, because you have the queue full, or it was that there is a rule that was configured to say that this packet needs to be dropped or you got a packet which was not having any matching rule and you do not have any means to process that packet further.

So, if we want to understand what are all the aspects that could be happening with a given flow packet for what purposes it was dropped and how it was being done, and if we consider it a switch, the most common would be where you have a queue and the queues building up and you want to know if the queue overflow happened or because you had a rule configured and you are trying to enter drop early on this particular packet.

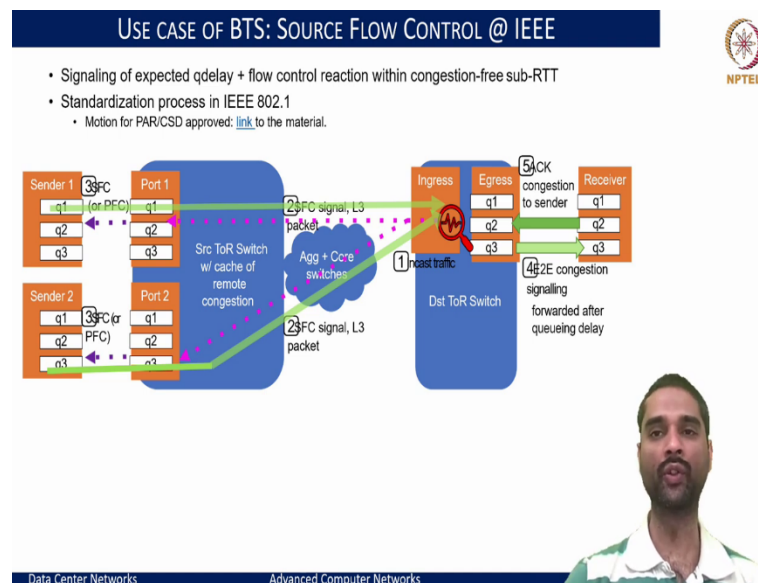
So, there may be different reasons why these packets could be dropped. So, if we see any such packets if they are dropped, we will want to understand precisely the reason and what triggered for this packet drop. And if we are able to configure a drop Watchlist into a switch and then whenever a packet that go through and if they were monitoring for these packets for if they are set the rule of the headers in the drop Watchlist.

And if such a packet becomes a victim, which is going to be dropped for either of the reasons we discussed, then information can be triggered out from the switch and sent to the monitor to report exactly the reason why the packets were dropped. And this would enable the network operator to understand what is going on, and what is the reason and behaviour, and take the course corrective measures.

And likewise, if we have a use case where you want to watch on the queue size, and if the queue size is indicative of congestion when it exceeds a certain length, we want to build this information and report back to the monitor, and in that way if the packet headers match it to be in the queue, Watchlist and see that there is a packet that has been queued for this duration, and then the information about the queue congestion can be reported back to the monitor with appropriate reasons of how much queue and in terms of what all packets accounted for that queuing at a particular switch, all of this information can be padded in the packet header and sent towards the monitor. And then this way, the network operators can again look that maybe this time there were these flows that were competing, and which resulted in this queuing.

And what is exactly the queuing behaviour, the way it is happening over the course of time can also be monitored thoroughly to see now if we want to retrofit and think about how we learned about the DCTCP the way it operates on the incast, is there a way that INT can have an impact over there.

(Refer Slide Time: 41:21)



And there was a demonstration that was done, which said, let us try to see how the incast happens and where this INT can really help. And what we saw, if you recollect, earlier, what we said, whenever multiple of the flows are being sent towards the destination, top of the rack switch or basically we call them as where the aggregators would see a lot of information from the worker nodes.

And the worker nodes are basically a lot of these at the top of the rack, which are now going to send the information towards the aggregator, and the aggregator is, again, somewhere in the other side of the top of the rack in a data center, towards which all of the other workers were trying to send the info packets.

And this all happens in a very small microsecond time scale, and we will end up seeing a queue build-up at the aggregator switch or the destination top of the rack switch. And this resulted in what we call as a TCP incast, packets will be dropped at the switch because of not having sufficient buffers for that microburst.

And the way we said we will react is basically we mark the packets as we forwarded them, whatever the packets are that instance, and send it and then send the receiver to send an acknowledgment back indicating the senders to slow down or change their sending behaviour.

And also, in DCTCP, we said it is going to be multibyte information that the senders will take into account and then adjust their rates accordingly. But one important observation that we have to make is such a thing to operate requires one round trip time, the time the congestion occurs, the information will go back to the sender only after the receiver has sent an ACK back with ECN.

And sender has to wait for one round trip to say how to react because he has to collect all the ECN bits, aggregate them and say how much was the congestion perceived, and then adapt the congestion window accordingly. So, if this can be bettered, and if we can leverage INT, what we can think of is to say that, at the ingress,, as the information is of the TCP incast and pursued congestion builds up, we can basically notify back directly as the packet info, the INT packet info, as a signal coming towards those particular senders at a time so that all the senders who are contributing towards the TCP incast or microburst that is resulting in congestion can all be notified upfront and make sure that this information reaches the sender much earlier. And this information can precisely cover what exactly has happened over a window of time within this ingress switch, and you do not need the senders to collect again and wait for RTT to take certain actions.

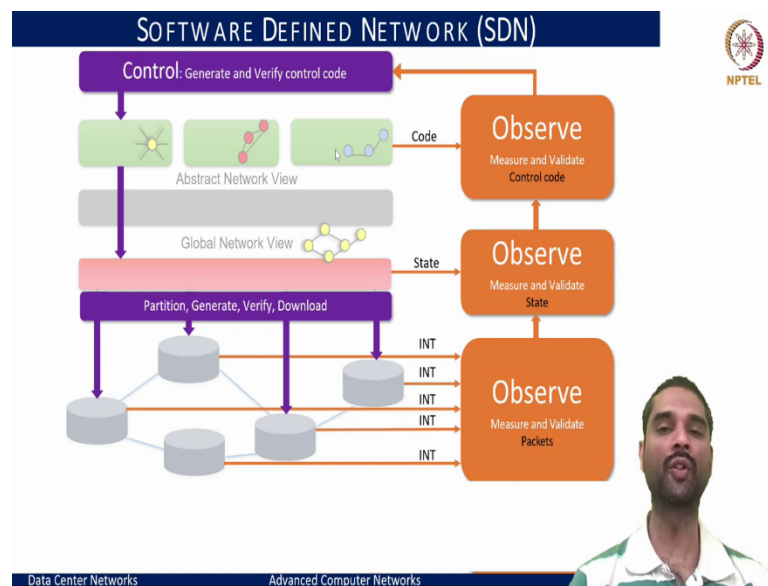
And that way, the information can be sent back in a much faster manner. And at the same time, once these switches receive this, they can apply either the source flow control or the priority flow control means to notify the sender and the sender can react back to say how they can react and slow down their sending rates.

And at the same time, as we did earlier with respect to how the congestion notifications were signed, that you see in the marking can still be followed and sent towards the receiver, and the receiver can then basically take this congestion notification signal and send the acknowledgement back to the sender.

And the sender can also have the traditional means of reaction that you will be doing, and overall now we have built a scheme where we are able to react much more faster, in a much more fine grained timescales than waiting for the entire round trip time.

And also leverage on the mechanisms to say how the senders should be able to react using either the SFC or PFC schemes between the source top of rack switches and sender and make this solution that can address the TCP incast problem. And this is also termed as back to sender or the BTS source flow control scheme in IEEE.

(Refer Slide Time: 45:33)



And to summarize, if we see now with the SDN word of what we started our discussions, at the data plane, we can have the inband network telemetry that can help monitor all the characteristics at each of the network elements. And this information can be collected and observed for any anomalies or any aspects that we would want to build analytics and pass it up to the control plane.

And the control plane as well can monitor this kind of state validate in terms of what is the behaviour and adapt its operations according to the monitored details that we collect from the INT. And this further can assist in enabling the pattern of how this control state changes are going to be impacted through the network management and operations.

So that now we have fine-grained control over the entire visibility of what is happening at the data plane, helping the control plane to make better decisions, and also the management plane in terms of the control applications to say what you want to run what kind of information you would want to build.

And all of this end to end observations can be built up, and this is where the other work that I mentioned from the IETF, the recent operations and management working groups specification in terms of IETF RFC 9232, which came just in around May 2022, speaks about the Applications of Network telemetry at different planes not just at the data plane like what the INT does.

Also applied at the control and management plane to ensure that you are able to build a framework that is holistic enough to get the information at each of the levels. For more details, you want to look at this IETF RFC 9232. But what we have covered is basic aspects of how the INT operates, what are its use cases, the operational modes, and the benefits that it brings to the table.