Ed Howorka Solutions

Working Paper – Categories: Trading Strategy, Trading Venue

# Colocation beats the speed of light

**Edward Howorka**
Principal, Ed Howorka Solutions
Published Feb 25, 2015 – Updated Apr 15, 2015

This paper demonstrates that traders gain nothing by positioning their computer at the midpoint between two financial exchanges – for example on a ship in the middle of an ocean – contrary to the recommendation in the article Physics in finance: Trading at the speed of light (Mark Buchanan, Nature, 11 February 2015). We show that, in fact, a trading strategy using cooperating computers colocated with the two exchanges will perform much better than a single computer located at the midpoint. This advantage of distributed trading strategies is non-obvious and – to my knowledge – has never been explained as simply and convincingly as in this paper. In the final section I examine the reverse issue: using distributed exchange server architectures that eliminate the need for distributed arbitrage strategies.

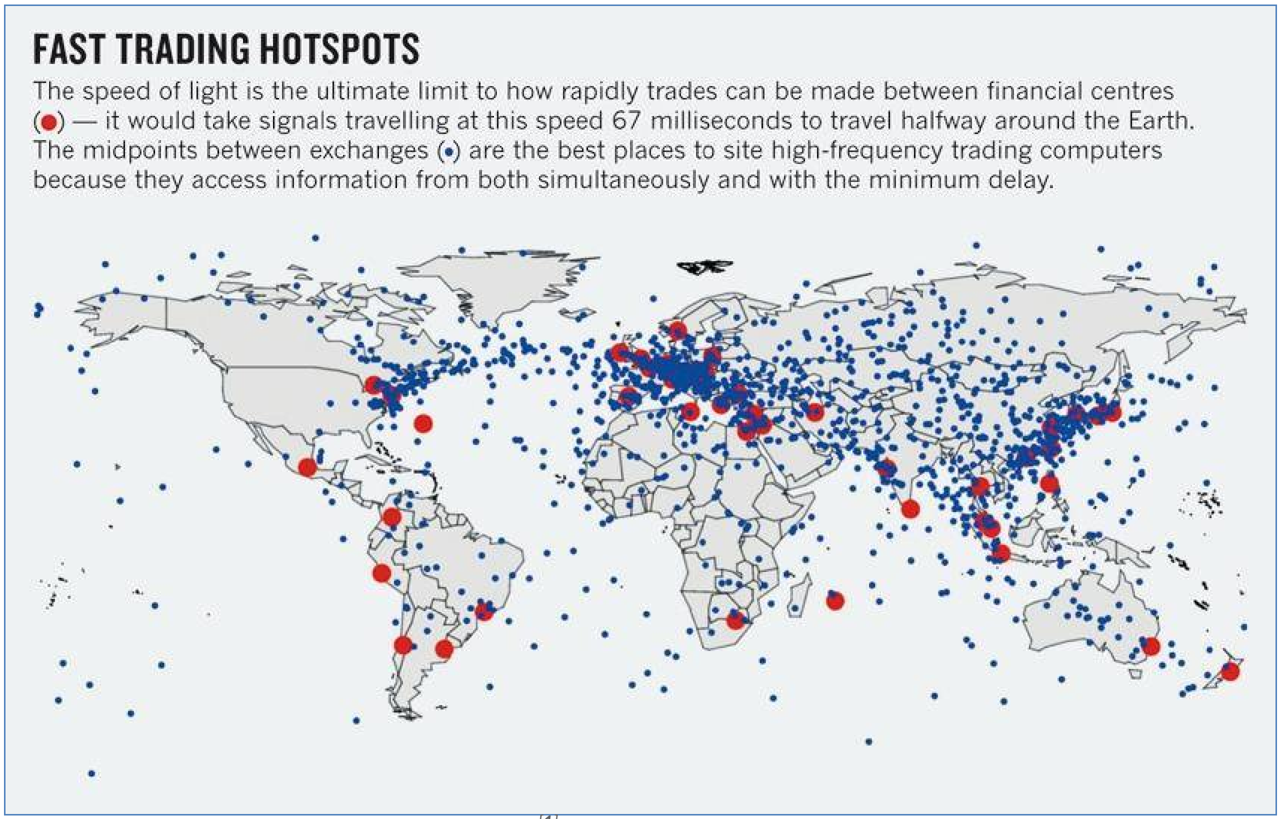
The Nature article states (emphasis mine):

> *In future, when airborne laser networks span the oceans, things may get even stranger. The location at which traders get the earliest possible information from two exchanges lies at their mid-point — between Chicago and London, this is in the middle of the Atlantic Ocean. At such a site, traders could exploit a technique called 'relativistic arbitrage' to profit from momentary imbalances in prices in Chicago and London.*

*To explain: special relativity says that nothing can travel faster than the speed of light, c. Hence, a trader standing a distance D away from an exchange can find out what happened there, in the best circumstance, at a time T = D/c after it happened. Between major trading centres around the globe, such delays can be from a few to tens of milliseconds. If a trader stands halfway between the two exchanges, he or she will receive information from both after the same interval, T = D/c. Anywhere else, the distance to at least one of the exchanges would be greater and information would take longer to get there.*

***In other words, within a few years it may become profitable to station a ship or other trading platform near halfway points between pairs of financial centres worldwide (see Fast-trading hotspots).***

The Nature article provides a map of the world showing all world financial centers and over a thousand midpoint locations where high-frequency traders would need to position their computers to optimally exploit the possible arbitrage possibilities. As expected, many of these "optimal" locations are truly in the "middle of nowhere" (one lies in the desolate spot where the flight MH 370 has disappeared). The map does not indicate the importance of the centers or the significance of the "fast trading hotspots", but it is clear that the most significant hotspots lie someplace in the North Atlantic Ocean.


**FAST TRADING HOTSPOTS**

The speed of light is the ultimate limit to how rapidly trades can be made between financial centres (●) — it would take signals travelling at this speed 67 milliseconds to travel halfway around the Earth. The midpoints between exchanges (•) are the best places to site high-frequency trading computers because they access information from both simultaneously and with the minimum delay.

*The map of Fast Trading Hotspots[1]. Used with permission of Nature Publishing Group.*

The map originally appeared in the paper Relativistic statistical arbitrage[2]. The "midpoints" in that article are weighted by turnover velocity (ratio of the value of shares traded on the exchange to the total market capitalization) – they are not true geodesic midpoints, as implied in the Nature article. Furthermore, the "financial centers" defined in the article turn out to be the 52 member stock exchanges listed in the World Federation of Exchanges 2009 Annual Report[3] – they do not include Forex exchanges which dwarf all other trading venues by an order of magnitude in terms of trading volumes.

The news regarding the optimal placement of HFT (high-frequency trading) computers (on ships!) was covered widely by professional mainstream media; see stories in PhysicsCentral[4], MarketWatch[5], International Business Times[6], and WatersTechnology[7]. Some pundits accepted the "fact" as self-evident, some expressed doubts. Bill Harts, CEO of Modern Markets Initiative, a group that supports high-frequency trading, dismissing the idea, told MarketWatch: *"Even if a trader could receive data from a market faster at such a location, in order to profit from it she would have to transmit orders to a market from the same location and would always be slower than traders stationed at the actual markets."*

## The midpoint puzzle

The Nature article is trivially correct in stating that the midpoint between the two exchanges is the point where an arbitrage opportunity can be first discovered. What is less obvious – given that Mark Buchanan, a respected physicist and author got it wrong – is that the midpoint location is a particularly *bad* place where to make trading decisions.

Here, Bill Harts' gut feeling is correct. In a follow-up IBT article[8], he states *"If this were possible, data center space in Youngstown, Ohio, (roughly half way between the New York and Chicago exchanges) would be at a premium because all the HFTs would locate there. But guess what? There is no HFT in Youngstown, Ohio."*

So, HFTs know that trading from a midpoint location between exchanges is a bad idea. However, Harts' statement is not entirely correct -- the midpoint trader is in fact as fast as anyone could be to take advantage of a price disparity between the two exchanges. Where she loses compared to "*traders stationed at the actual markets"* is that – after submitting her orders, she is blind and out of control of her orders. Read on to see the truly incredible advantage of server colocation and distributed trading strategies.

## The approach

For the sake of generality we will refer from now on to "*servers*" rather than "exchange servers", because our result applies to scenarios much more general than a trading algorithm communicating with financial exchanges. We will use the word "*computer*" to refer to a non-server computer running a program that communicates with the servers (for example, a trading strategy algorithm).

In the next few sections we will prove the following fundamental theorem. (Please do not be concerned -- the proof will be as easy as constructing a gadget from Lego blocks.)

**Theorem 1**. Any algorithm that runs on a single central computer located anywhere in the world and is connected to $N \geq 1$ servers can be re-implemented using a group of $N$ computers that are colocated with the respective servers in such a way that the new implementation's behavior (as observed by the servers) is indistinguishable from the original single computer solution.

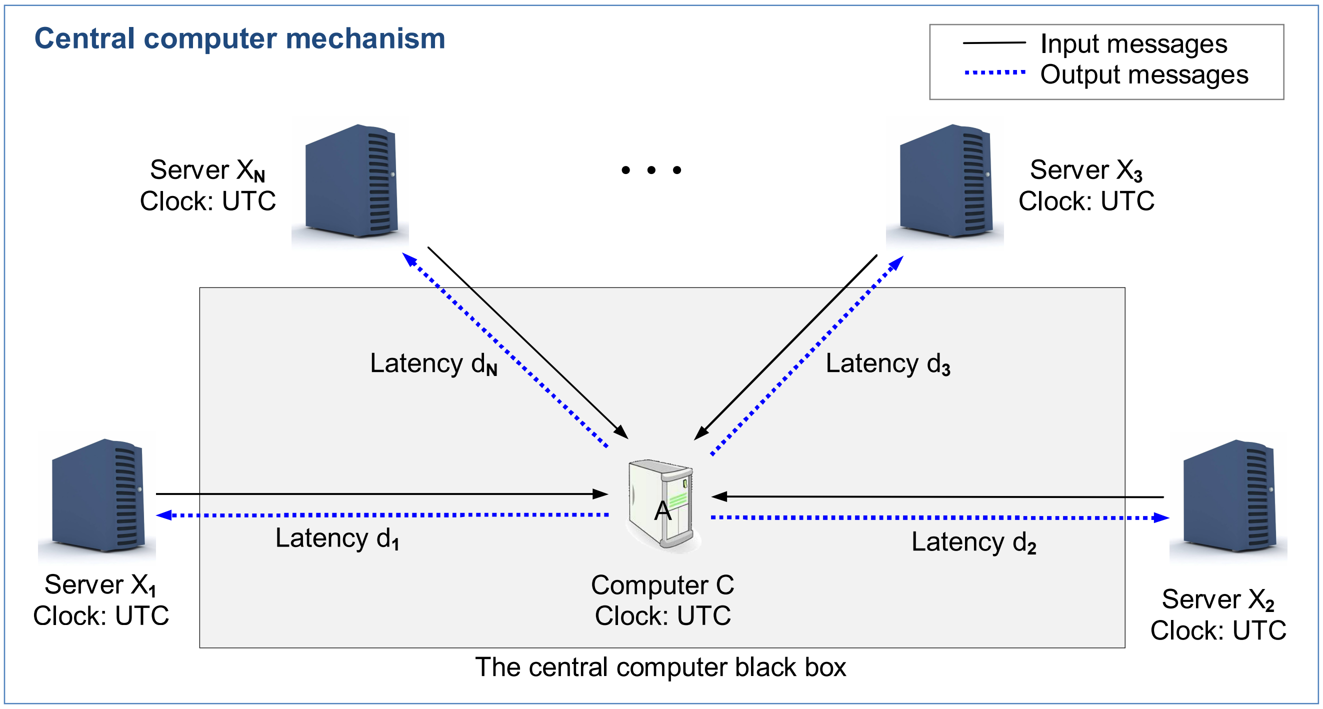
The proof is quite straightforward. First, we formally define the single central computer mechanism. We then show how to construct an equivalent mechanism that uses a group of computers colocated with the respective servers, with no need for the central computer. In other words, we start with a black box implemented using a central computer and show how to construct an equivalent black box without a central computer.

Finally, we will point out the obvious superiority of the distributed solution.

## The central computer mechanism

First, let us define formally the basic "central computer mechanism". Note that it is much more general than the midpoint computer scenario considered in the Buchanan's article.

Consider *N* servers $X_1$, …, $X_N$ and a central computer *C* located anywhere in the world and running an algorithm *A* (for example, a trading strategy). *C* is connected to all of the servers – it receives inputs from and sends outputs to each of them. We think of this mechanism as a black box where the inputs and outputs can be observed and timed by observers located at the *N* servers. See the Central computer mechanism figure.



The central computer *C* and each of the *N* servers have a clock that is synchronized to the UTC time. (Today's technology provides means to synchronize clocks to within a small fraction of a microsecond.)

Let $d_i$ denote the transmission latency between *C* and the server $X_i$. Let $d_{i,j}$ denote the latency between $X_i$ and $X_j$. These latencies are assumed to be constant and not to vary from message to message. We assume that the network latency satisfies the properties of a metric, in particular: $d_{i,j} = d_{j,i}$ (symmetry) and $d_{i,j} \leq d_i + d_j$ (triangle inequality).

In order to perform our black box transfiguration (from the central computer to an equivalent group of computers colocated with the servers) we need to make certain additional assumptions:

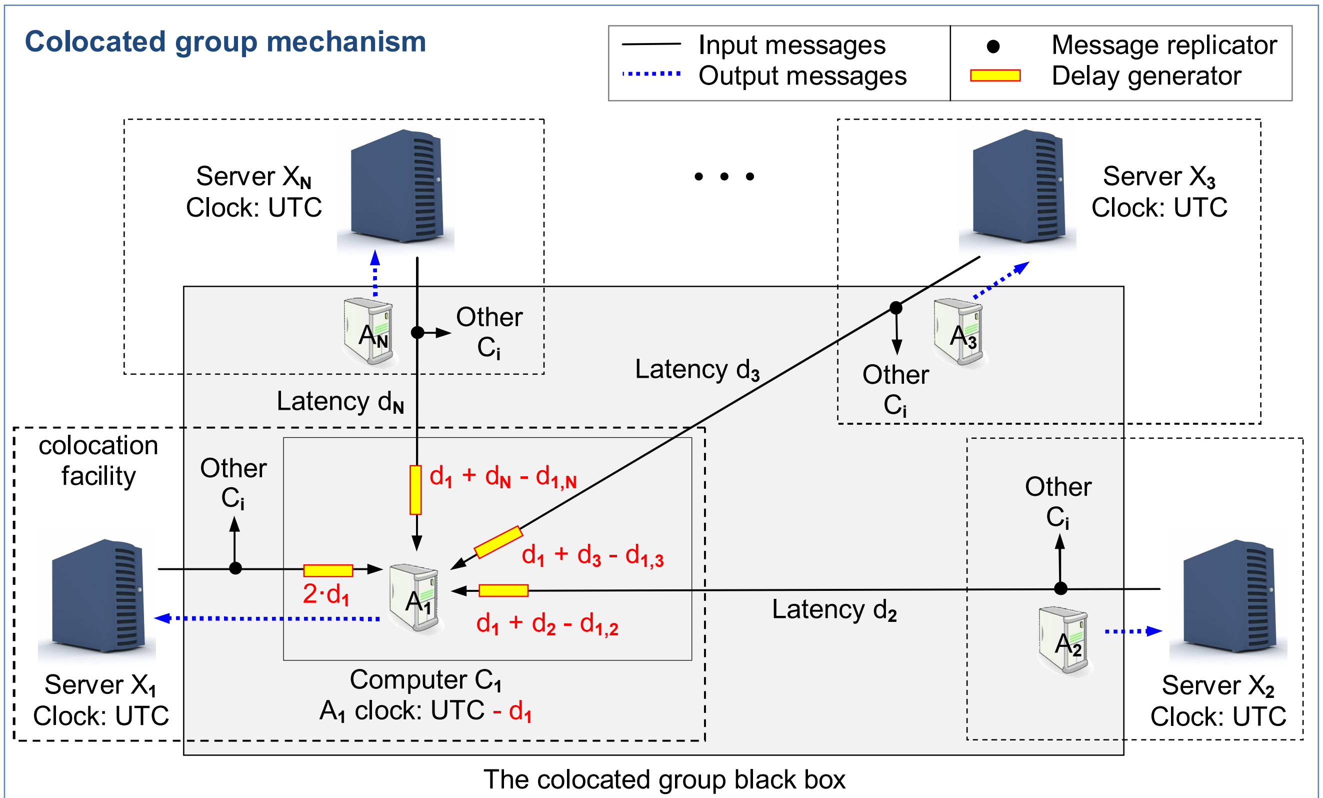
1. Once configured, the algorithm *A* is driven only by the current time and by messages received from servers $X_i$. (Of course, the algorithm *A* can also use its memory of past events in making its decisions.)
2. Each of the *N* servers sends time-clocked streams of data with defined time granularity. This means that the listener receives periodic synchronization signals from each server that define and close each time interval – whether it is a millisecond or a microsecond. The algorithm *A* is driven by these signals. (For example, a message received at time *t* by the algorithm *A* can be

used by *A* only after all the time synchronization signals time stamped with time greater than or equal to *t* - $d_i$ are received from all servers $X_i$. This is not a hindrance, as this will happen naturally in the central computer scenario. We need this requirement only to guarantee the equivalency of our black boxes in spite of any time or latency jitter.)

3. The output of the algorithm *A* consists of messages sent to the servers. All messages are time stamped using the local clock used by *A* and may contain any information, including internal status reports referring to the current state or to past events.

## The colocated group mechanism

Here, we present a mechanism that will duplicate the behavior of the central computer mechanism using *N* computers $C_1$, ..., $C_N$ colocated respectively with the servers $X_1$, ..., $X_N$. Just as in the previous case, we think of this mechanism as a black box where the inputs and outputs can be observed and timed by observers located at the *N* servers. See the Colocated group mechanism figure.



This figure shows how the computer $C_i$ (collocated with the server $X_i$) is connected to all the exchange servers. Only the $X_1$ colocation setup is shown in detail. The setups for $X_2$, …, $X_N$ are analogous.

For the sake of simplicity, the transmission latency between the computer $C_i$ and the colocated server $X_i$ is taken to be zero. The transmission latency between $C_i$ and $X_j$ is taken to be $d_{i,j}$ – the same as the latency between $X_i$ and $X_j$.

Each computer $C_i$ runs an algorithm $A_i$ which is a modified copy of the algorithm *A* running on the central computer *C* described in the last section. Here are the modifications:

1. The clock used by $A_i$ is set back by $d_i$, that is, it shows time UTC - $d_i$ instead of UTC. For example, when the algorithm $A$ decides to stop trading at UTC time $t$, $A_i$ would stop trading at UTC time $t + d_i$. (It is convenient to think of this as a modification of the algorithm $A_i$, rather than a change of the clock of $C_i$.)
2. Each algorithm $A_j$ receives messages from *all* N servers. Specifically, the messages from a server $X_i$ that would be sent to the original central algorithm $A$ are replicated and sent to all algorithms $A_j$. Message replication may be accomplished either in hardware or in software running on the computer $C_i$ colocated with $X_i$.
3. An artificial delay $d_i + d_j - d_{i,j}$ is introduced for all messages received by $A_j$ from the server $X_i$. Message delays may be implemented either in hardware or in software running on $C_j$. (It is convenient to think of the delay generators as a modification of the algorithm $A_j$.) Note that when $i = j$, the delay is $d_i + d_i - d_{i,i} = 2 \cdot d_i$. That is, the delay for all messages sent by the server $X_i$ to the collocated computer $C_i$ is $2 \cdot d_i$.
4. The algorithm $A_i$ sends its output messages only to the colocated $X_i$ server. When the original algorithm $A$ would send a message to another server $X_j$, $A_i$ does nothing – instead, the algorithm $A_j$ colocated with $X_j$ sends the identical message at precisely the right time. $A_i$ may log this event, however, just as $A$ would do so.

## The central computer and the colocated group mechanisms are equivalent

To verify that the colocated group mechanism works the same as the central mechanism, one needs to verify that given inputs result in the same outputs, received at the same time by the servers, for both black box mechanisms.

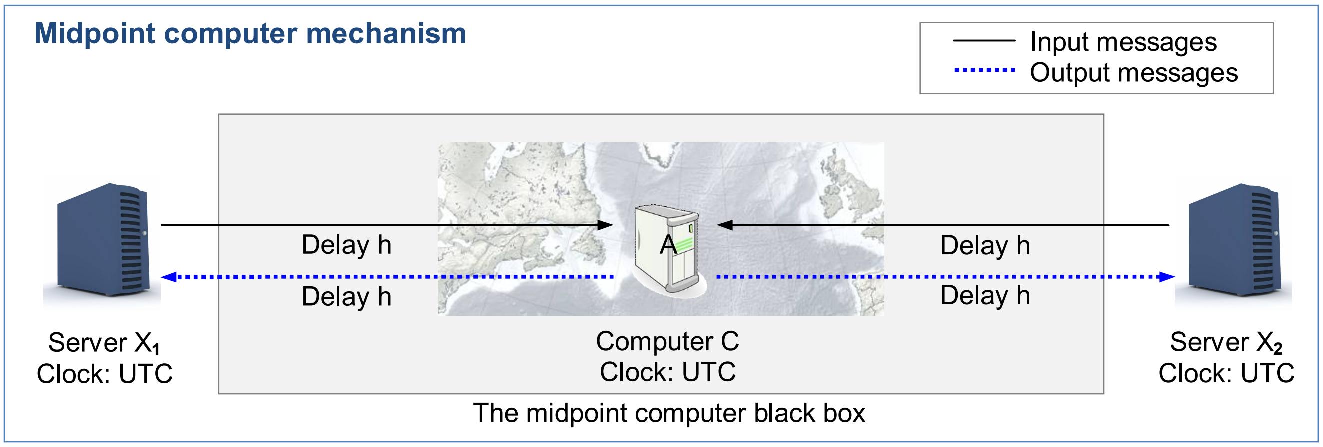
It is easy to see that, **for any *j***, the algorithm $A_j$ receives exactly the same message inputs as would be received by the algorithm $A$, but $A_j$ receives them after a delay $d_j$ with respect to the clock used by $A$. Specifically, at time $t$, $A$ would receive a message from server $X_i$ that was sent at time $t - d_i$ . That same message is sent from $X_i$ to $A_j$ through an artificial delay generator ($d_i + d_j - d_{i,j}$) and then through a transmission line that has latency $d_{i,j}$, thus arriving at $A_j$ at time $(t - d_i) + (d_i + d_j - d_{i,j}) + d_{i,j} = t + d_j$.

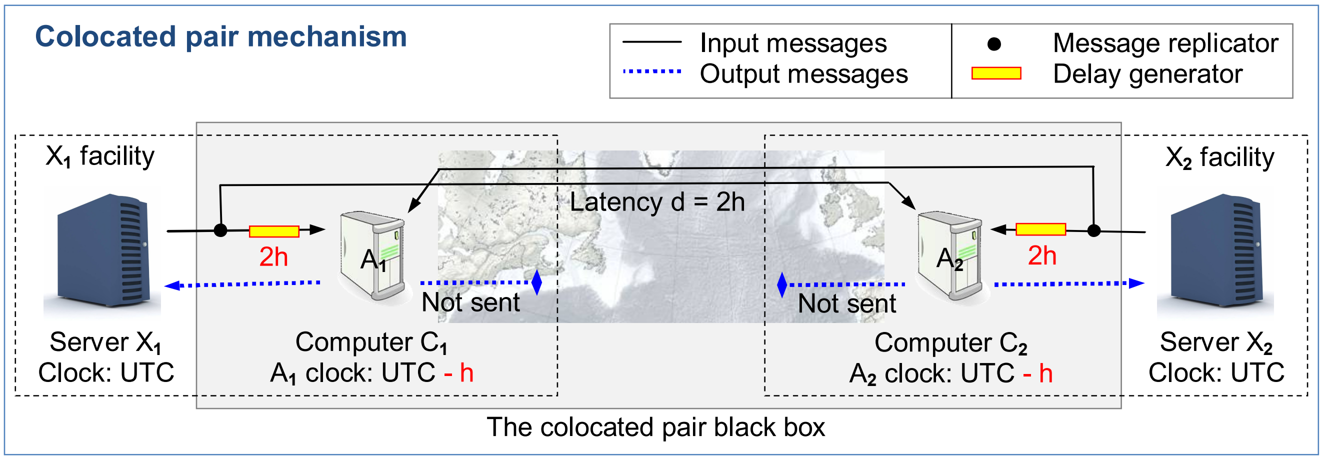
The delay $d_j$ is rectified by setting the clock of $A_j$ back by $d_j$. This means that, at UTC time $t + d_j$, all inputs of $A_j$ – including the clock – are identical to those of $A$ at UTC time $t$. Therefore, $A_j$ makes the same decisions at time $t + d_j$ as $A$ would make at time $t$. Although the decisions made by $A_j$ are made with the delay $d_j$, its outputs are delivered instantly to the server $X_j$ (without the delay $d_j$ suffered by outputs of $A$ sent to $X_j$), which means that both the timestamps affixed by $A_j$ and the message delivery times observed by $X_j$ are identical to the ones produced by the algorithm $A$. This proves Theorem 1.

## An example

To get a feel for how the colocated group contraption works, we will use the example from the Nature magazine article. Here, we start with a single "midpoint computer" $C$ located at the geodesic midpoint between two financial center servers $X_1$ and $X_2$ (think of NY and London and imagine trading EUR/USD in the Forex spot market), see the Midpoint computer mechanism figure.

Using simpler notations, $\boldsymbol{d} = d_{1,2}$ (= about 20 milliseconds at the speed of light) and $\boldsymbol{h} = d_1 = d_2 = d/2$, and using the general method presented above, we construct a simple functionally equivalent mechanism consisting of a pair of computers colocated with the two exchanges shown on the Colocated pair mechanism figure.





Next, we illustrate a concrete arbitrage strategy algorithm in action, tracing it step-by-step in both scenarios. Our midpoint arb strategy algorithm *A* is defined as follows: If the ask price on one side (say, $X_1$) is strictly lower than the bid price on the other side ($X_2$), then send a limit FOK (fill-or-kill) *buy* order to $X_1$ and a limit FOK *sell* order to $X_2$. If both orders succeed, we are done and we made a profit. Otherwise (if, say, the sell on $X_2$ failed), we retreat by liquidating our position using a market order sent to the exchange that offers the better bid price. (This is not the smartest arbitrage algorithm, it is just involved enough to illustrate what's going on inside the two black boxes.)

| **UTC** | **Strategy algorithm *A*** | | **Strategy algorithm $A_1$** | **Strategy algorithm $A_2$** |
|---|---|---|---|---|
| $t_0$ | An arbitrage opportunity arises: the ask on $X_1$ is lower than the bid on $X_2$. | | | |
| $t_0+1h$ | Looking at the $X_1$ and $X_2$ prices time stamped $t_0$, *A* discovers the arb opportunity. *A* sends a FOK buy to $X_1$ and a FOK sell to $X_2$. | | | |
| $t_0+2h$ | $X_1$ and $X_2$ servers receive the orders sent by *A*. The buy sent to $X_1$ succeeds, but the sell sent to $X_2$ fails. | | Looking at the $X_1$ and $X_2$ prices time stamped $t_0$, $A_1$ discovers the arb opportunity. $A_1$ sends a FOK buy to $X_1$. The buy succeeds. | Looking at the $X_1$ and $X_2$ prices time stamped $t_0$, $A_2$ discovers the arb opportunity. $A_2$ sends a FOK sell to $X_2$. The sell fails. |

| | | | | |
|---|---|---|---|---|
| $t_0+3h$ | *A* receives order status messages from both servers. It decides to liquidate the position by selling to the best bid which happens to be on $X_2$. *A* sends a market sell to $X_2$. | | | |
| $t_0+4h$ | The market sell order sent by *A* is executed by the $X_2$ server. | | $A_1$ receives order status messages from both servers. It decides to liquidate the position by selling to the best bid which happens to be on $X_2$. $A_1$ does nothing (knowing that $A_2$ will execute the market sell). | $A_2$ receives order status messages from both servers. It decides to liquidate the position by selling to the best bid which happens to be on $X_2$. $A_2$ sends a market sell to $X_2$ which is executed by the $X_2$ server. |
| $t_0+5h$ | *A* receives the order status message from $X_2$ and updates its trading balance. The balance with the timestamp "$t_0+5h$" is sent to observers on both sides ($X_1$ and $X_2$). | | | |
| $t_0+6h$ | The balance with the timestamp "$t_0+5h$" is received by observers on both sides ($X_1$ and $X_2$). | | $A_1$ receives the order status message from $X_2$ and updates its trading balance. The balance with the timestamp "$t_0+5h$" is sent by $A_1$ and received by the observer on the $X_1$ side. | $A_2$ receives the order status message from $X_2$ and updates its trading balance. The balance with the timestamp "$t_0+5h$" is sent by $A_2$ and received by the observer on the $X_2$ side. |

## Colocation provides a better solution for high-speed applications

We have shown that the central computer mechanism can be replaced by a *functionally identical* mechanism using a group of computers colocated with the servers. Paradoxically, in order to achieve the equivalent behavior we had to make use of grotesque artificial delay generators that delay information transmission from each server to the colocated computer by the full round-trip latency *d* between the server and the central computer (plus additional inter-server delays in the general colocated group mechanism). For most central computer algorithms, removing the delay generators in the collocated scenario would allow the algorithm designers to develop a distributed algorithm that is not only equivalent to, but far better than the central computer algorithm. The nature and the significance of the improvements would depend on the algorithm and its function. In most cases, the central computer algorithm would have to be rewritten from scratch to take advantage of the foreknowledge of the local market. (And, in some cases, no improvement would be possible. Consider, for example, hourly blind auctions of some sort; here, using one central computer would be equivalent, optimal, and more economical than a group of colocated computers.)

Imagine, in our NY-London colocated pair example from the last section, having to slow down information delivery at a NY data center by a factor of 10,000 (from the typical two microseconds to 20 milliseconds). No self-respecting high-speed trader would accept such design. Removing the delay generators would enable the colocated computers to "see the local future" by comparison with what the central computer sees; this would allow development of a vastly superior trading strategy. A high-frequency trading firm would pay millions of dollars per month to see 20 ms into the future. In other words, if the computer located at the midpoint between financial exchanges represented the best way to cope with the speed of light limitations, then using computers colocated with the exchanges "beats the speed of light" – whence the title of this paper.

But even *without* any additional improvements, the collocated strategies need to be deployed in only *N* financial centers worldwide, not the *N·(N-1)/2* "fast trading hotspots" shown in the Nature article. Finally – and best of all – we will never need the scary high-speed trading ships!

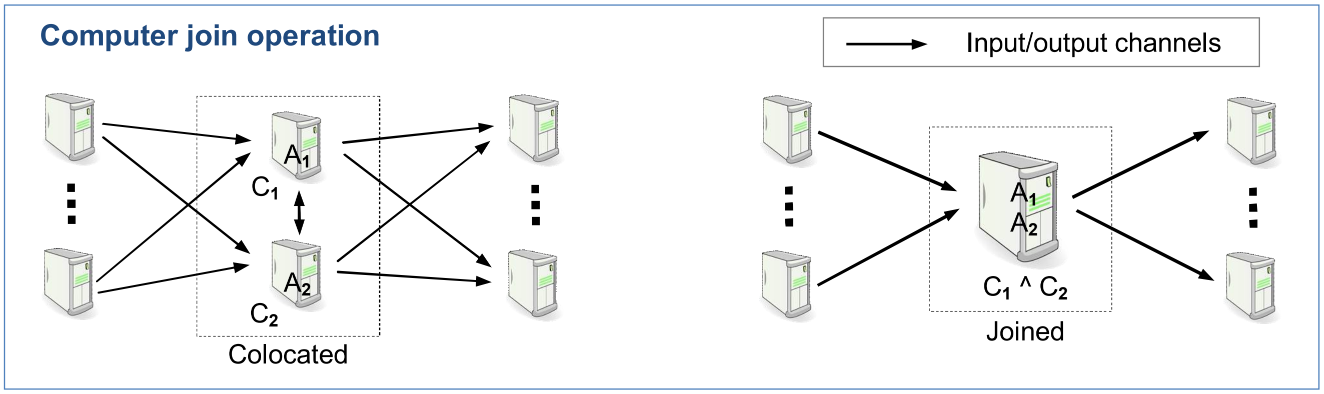
## The fully collocated solution can emulate any other solution

We have shown that a single central computer mechanism can be always be replaced by a *functionally equivalent or better* mechanism using a group of computers colocated with the servers. One question that remains is: Can that solution be improved? Could we add a few extra computers in the middle and/or remove some of the colocated computers and get a *strictly better* solution than any solution that could be implemented using the fully colocated group? The answer is ***no***. The full group of *N* computers colocated with the *N* servers provides the optimal configuration which cannot be improved.

In order to state this formally we define a new term. An *aggregate* is a group of computers running a distributed algorithm, including all communication channels between the members of the group and all input and output channels to the outside world (consisting of the "servers"). Unlike for a single computer running an algorithm, an aggregate specification must include details concerning spacial locations of the inputs and outputs as well as the member computers, and latency constraints on communications between these locations. An aggregate is a more formal term for what we called a "black box" in the preceding sections.

Given *N* servers, a *fully colocated aggregate* is an aggregate of N computers colocated with the respective servers.

Finally, to proceed with our proof, we need to define a computer "join" operation: if $C_1$ and $C_2$ are two single colocated computers running algorithms $A_1$ and $A_2$, respectively, then $C_1$ ^ $C_2$ denotes the computer that runs the same algorithms and uses the combined input/output channels of the two computers as its input/output channels, respectively (see the Computer join operation figure). The join operation is purely cosmetic. The two algorithms run independent of each other on the join computer and retain all of their attributes (such as a clock offset and input delays, etc.). The messages received and sent by two algorithms remain the same as well. The join operation allows us to consider two colocated computers as a single computer that is functionally equivalent to the pair being joined. The operation is commutative and associative.



We are now ready to state and prove our main result.

**Corollary 1**. Any aggregate connected to a number of servers can be re-implemented using a fully colocated aggregate in such a way that the new aggregate's behavior (as observed by the servers) is indistinguishable from the original aggregate.

The proof uses mathematical induction on the number $K$ of members of the aggregate that are not colocated with any of the servers. Here is the outline.

It is easy to see that the corollary is true for $K = 0$. Here, each member of the aggregate is colocated with one of the servers. In order to form an equivalent fully colocated aggregate, we add do-nothing aggregate members to servers that have no colocated member, and we replace all aggregate members colocated with a given server with their join to ensure a single aggregate member per server.

Assume now that the corollary is true for certain $K \geq 0$.

Consider an aggregate $D^*$ connected to $N$ servers ($X_1, \ldots, X_N$) that has $K+1$ members that are not colocated with any of the servers.

We begin by normalizing $D^*$: we add do-nothing aggregate members to servers that have no colocated member of $D^*$, and we replace all aggregate members colocated with a given server with their join to ensure a single aggregate member per server. This does not change the number of non-colocated members of $D^*$ or the aggregate's behavior.

From now on, we assume that $D_i$ is the single member of $D^*$ colocated with the server $X_i$ for $i = 1, \ldots, N$. and that the members $D_{N+1}, \ldots, D_{N+K+1}$, are not colocated with any of the servers.

Finally, we assume that the member $D_{N+K+1}$ is not directly connected to any server $X_i$. If it was connected directly to $X_i$ , we could always task the collocated computer $D_i$ to become a pass-through intermediary between $X_i$ and $D_{N+K+1}$.

We now consider all of the $D_1, \ldots, D_{N+K}$ computers as the "servers" in Theorem 1. Let $C = D_{N+K+1}$ be the central computer communicating with the "servers" $D_1, \ldots, D_{N+K}$. By Theorem 1, we can construct a functionally identical system that replaces $C$ with $N+K$ computers $C_i$ colocated with the respective computers $D_i$.

Finally, define a new aggregate $E^*$ running on computers $E_1, \ldots, E_{N+K}$ where we define $E_i = C_i \wedge D_i$.

It is easy to see that the aggregate $E^*$ is equivalent to $D^*$ (when observed by the servers $X_i$), but has only $K$ members that are not collocated with any server. Therefore, by the inductive assumption, $E^*$ is equivalent to a fully colocated aggregate. This completes the inductive step and proves the corollary.

**Note**: The proof of the corollary above may be viewed as providing a technique that allows one to eliminate one-by-one any non-colocated members of the aggregate $D^*$ without changing the aggregate functionality (as observed by the servers).

## Colocation provides the optimal solution

In order to discuss "optimality" we assume only that the user has in mind a numerical "measure of goodness" that can be applied to any aggregate interacting with a predefined set of servers. We do assume that the "goodness" function is based only on what's observable by the servers. (For example, one could evaluate a trading strategy by its average daily profits based on historical benchmark data collected by the servers.) We can now rephrase Corollary 1 to state that:

**Corollary 2**. The fully collocated aggregate architecture is optimal. In other words, no distributed algorithm can beat all algorithms implemented using a fully collocated aggregate.

**Note**: The corollary above is remarkably general, considering our lenient definition of what "better" means. No matter how you measure the performance of your algo (for example, whether you desire to make or to lose more money), you cannot beat the distributed setup. And, of course, you should be particularly leery of "central computer" solutions. As an example: Do not run a yen intervention centrally out of Tokyo (or even out of London, for that matter)!

## Reversing the roles - distributed exchanges beat the speed of light

So far, we considered Theorem 1 in scenarios consisting of a trading computer connected to many exchange servers, because that scheme fits the midpoint computer scenario in the Nature article. We will now restate Theorem 1 by reversing the roles of exchanges and trading centers.

Traditional exchanges have a central matching engine with HFT customers' computers colocated in the same data center. Remote customers may be distributed throughout the world using computers located in *access point data centers* (provided either by the exchange itself or by third-party companies) that offer secure, reliable, and consistently low-latency connectivity to the central engine.

Let us now consider an exchange's matching engine as the "central computer" and the access point data centers as the "servers" cited in Theorem 1. Using this sleight of hand, Theorem 1 can be restated as follows:

**Corollary 3**. Any single central exchange matching engine connected to $N \geq 1$ access point data centers can be re-implemented using a group of $N$ interconnected matching engines that are located in the respective data centers in such a way that the new implementation's behavior (as observed by the data centers) is indistinguishable from the original single matching engine solution.

Corollaries 1 and 2 may be restated in a similar manner. In particular, exchange architecture that uses a fully collocated aggregate of matching engines (one matching engine per access point data center) is optimal!

## The case for distributed exchanges

Obviously, HFTs figured out long ago the advantages of colocation when arbing AAPL between different NJ exchange servers or when arbing E-Mini on CME in Aurora, IL versus SPY on Arca in Secaucus, NJ. So, why do exchanges persistently use single central matching engines?

The main reason is exchange's performance and profits. By comparison with a single central exchange computer, distributed exchange architecture incurs penalties in terms of design and behavior complexity, increased transaction latency, and additional infrastructure costs. These effects would be experienced and borne by all customers, even those that do not engage in arbitrage and are happy to trade in a single location, colocated with a simple, fast, and relatively inexpensive central matching engine. Distributed exchange architecture would also reduce (or eliminate) the need for arbitrage and the resulting massive trading volumes that benefit the exchange.

In the US stock market, trading is geographically clustered in one location. Starting with NYSE and Nasdaq, practically all stock trading in US takes place on servers in New Jersey that are within a few tens of miles from each other. Although (thanks to 1998 SEC Regulation ATS) the same stock may be traded on several exchanges, each exchange usually has only a single matching engine for a given stock symbol. In spite of all this, one could still argue that, overall, the US equities market uses a distributed order matching architecture of sorts:

1. Equities exchanges and ATS (alternative trading systems) are required to route orders to other venues to obtain the best price (thanks to 2007 SEC regulation NMS).

2. Smart order routers located at the access points send the order to the exchange where the order has the greatest chance of getting done at the best possible price.
3. Broker/bank internalization engines and dark pools could be considered as extensions of the primary exchange matching machinery.

Of course, the devil is in the detail, and when one looks closely, the existing US equities exchange architecture (when considered in its totality as a single mechanism) is horrendously complex and inefficient. HFTs' computers colocated with multiple proximate exchanges prosper using arbitrage and other clever techniques to exploit the inefficiencies.

In the equities futures and options markets, for historical reasons going back to trading futures in corn or wheat, most US stock futures and options trade in Chicago (on CME and CBOE, respectively). Here, it would be logical to simply move the CME and CBOE matching engines to New Jersey. But this is unlikely to happen given that there are (yet) no competitive pressures on the Chicago exchanges to do so. HFTs seem to be happy competing with each other to build better-cheaper-faster distributed arbitrage strategies (using the Spread Networks tunnel and its true-speed-of-light successors[9]).

## Truly distributed exchange architecture

Finally, let us ask the opposite question: Why and where would one need distributed exchange servers matching orders for a specific instrument in distant locations? One answer is that distributed exchanges would be useful if customers were geographically “distributed” and unwilling to colocate their trading computers with the exchange’s central order matching engine.

So far, the only area where distributed exchange architecture found its justification is Forex trading. For example, when you trade USD (US dollar) against JPY (Japanese yen), you need local information about the state of economy of each of the two very distant countries. In addition, there are political reasons against trading one’s currency using a matching engine in a faraway overseas location.

When the Forex spot electronic order matching was first introduced in the 1990’s, there were initially two USD/JPY exchanges: Reuters (based in London) and Minex (based in Tokyo). London-Tokyo latencies in those days were on the order of 300 ms, easily noticeable even by human traders. When I was tasked with designing a new global EBS Forex exchange, I chose a distributed architecture with three synchronized order matching engines – in London, NY, and Tokyo[10]. It provided excellent performance in all three regions and satisfied the priorities of the global banks. In 1996, EBS merged with – and replaced – Minex, becoming the dominant global USD/JPY interbank electronic exchange; the position that it maintains to this day. (I wish I had the insights of this paper when trying to convert my colleagues to the belief in distributed exchange architecture… but in the end, we used our gut feeling – and we were proven right.)

Another example of a distributed Forex exchange is Fastmatch, which also has three matching engines in London, NY, and Tokyo. Its architecture differs significantly from that of EBS, however, as orders are not transparently propagated and matched across the three regions.

## Further reading

To find more about the wonders of distributed trading systems, study references [11] and [12].

## Acknowledgments

I thank Michael Merold, Schalk Steyn, Alec Schmidt, Mark Reece, Wlodek Stankiewicz, Ewa Howorka, and others who have provided helpful comments on the initial draft of this paper.

## References

1. Mark Buchanan, Physics in finance: Trading at the speed of light, Nature, Feb 11, 2015.
2. A. D. Wissner-Gross and C. E. Freer, Relativistic statistical arbitrage, Physical Review E 82, Nov 5, 2010.
3. World Federation of Exchanges, WFE 2009 Annual report and statistics, Dec 2009.
4. Stock Trades at the Speed of Light, PhysicsCentral, Nov 11, 2010.
5. Eric Garcia, Physicist says it is possible to use ships to facilitate high-frequency trading, MarketWatch, Feb 19, 2015.
6. M. Rohan, Dark pool and HFT: It is possible to use ships for high-frequency trading, International Business Times, Feb 19, 2015.
7. Timothy B. Murray, HFT's Oceanic Moment, WatersTechnology, Mar 11, 2015.
8. M. Rohan, Dark pool and HFT: BoE says high-speed trading helps markets, International Business Times, Feb 24, 2015.
9. Scott Patterson, High-Speed Stock Traders Turn to Laser Beams, Wall Street Journal, Feb 11, 2014.
10. Michael Togher, Michael F. Dunne, Richard Hartheimer, Credit management for electronic brokerage system, U.S. Patent No. 5,375,055, issued Dec 20, 1994, assigned to EBS.
11. Edward R. Howorka, Andrew P. Foray, Architecture for anonymous trading system, U.S. Patent No. 7,184,982, issued Feb 27, 2007, assigned to EBS.
12. Edward R. Howorka, Method and apparatus for enhancing market data feed using proprietary order flow, U.S. Patent No. 8,296,217, issued Oct 23, 2012, assigned to MarketFactory.
13. Edward Howorka, In brief, colocation beats the speed of light, The Trading Mesh, Apr 9, 2015.